Vilnius Gediminas Technical University

Jelena Mamčenko

Lecture Notes

On

OPERATING SYSTEMS

Code **FMITB01001**
Course title **Operating Systems**
Course volume **4,0 cr. (6,00 ECTS cr.)**
Teaching methods (Full-time, daytime studies):
Lectures - **32 h per semestre**
Laboratory works - **32 h per semestre**
Individual work - **96 h per semestre**
Assesment - **Examination**

Course aim:
Understandig of Operating System's functions.

Course description:
Concept of operating system. Control the activities and resourses of computer. Interpreting comands. Coordinating activities. Operating systems MS-DOS, OS/2, UNIX. Networking.

CONTENT

1 Introduction

Modern general-purpose computers, including personal computers and mainframes, have an operating system to run other programs, such as application software. Examples of operating systems for personal computers include Microsoft Windows, Mac OS (and Darwin), Unix, and Linux.

The lowest level of any operating system is its kernel. This is the first layer of software loaded into memory when a system boots or starts up. The kernel provides access to various common core services to all other system and application programs. These services include, but are not limited to: disk access, memory management, task scheduling, and access to other hardware devices.

As well as the kernel, an operating system is often distributed with tools for programs to display and manage a graphical user interface (although Windows and the Macintosh have these tools built into the operating system), as well as utility programs for tasks such as managing files and configuring the operating system. They are also often distributed with application software that does not relate directly to the operating system's core function, but which the operating system distributor finds advantageous to supply with the operating system.

The delineation between the operating system and application software is not precise, and is occasionally subject to controversy. From commercial or legal points of view, the delineation can depend on the contexts of the interests involved. For example, one of the key questions in the United States v. Microsoft antitrust trial was whether Microsoft's web browser was part of its operating system, or whether it was a separable piece of application software.

Like the term "operating system" itself, the question of what exactly should form the "kernel" is subject to some controversy, with debates over whether things like file systems should be included in the kernel. Various camps advocate microkernels, monolithic kernels, and so on.

Operating systems are used on most, but not all, computer systems. The simplest computers, including the smallest embedded systems and many of the first computers did not have operating systems. Instead, they relied on the application programs to manage the minimal hardware themselves, perhaps with the aid of libraries developed for the purpose. Commercially-supplied operating systems are present on virtually all modern devices described as computers, from personal computers to mainframes, as well as mobile computers such as PDAs and mobile phones.

2 History of Operating Systems

An **operating system** (OS) is a software program that manages the hardware and software resources of a computer. The OS performs basic tasks, such as controlling and allocating memory, prioritizing the processing of instructions, controlling input and output devices, facilitating networking, and managing files.

The first computers did not have operating systems. However, software tools for managing the system and simplifying the use of hardware appeared very quickly afterwards, and gradually expanded in scope. By the early 1960s, commercial computer vendors were supplying quite extensive tools for streamlining the development, scheduling, and execution of jobs on batch processing systems. Examples were produced by UNIVAC and Control Data Corporation, amongst others.

Through the 1960s, several major concepts were developed, driving the development of operating systems. The development of the IBM System/360 produced a family of mainframe computers available in widely differing capacities and price points, for which a single operating system OS/360 was planned (rather than developing ad-hoc programs for every individual model). This concept of a single OS spanning an entire product line was crucial for the success of System/360 and, in fact, IBM's current mainframe operating systems are distant descendants of this original system; applications written for the OS/360 can still be run on modern machines. OS/360 also contained another important advance: the development of the hard disk permanent storage device (which IBM called DASD). Another key development was the concept of time-sharing: the idea of sharing the resources of expensive computers amongst multiple computer users interacting in real time with the system. Time sharing allowed all of the users to have the illusion of having exclusive access to the machine; the Multics timesharing system was the most famous of a number of new operating systems developed to take advantage of the concept.

Multics, particularly, was an inspiration to a number of operating systems developed in the 1970s, notably Unix. Another commercially-popular minicomputer operating system was VMS.
The first microcomputers did not have the capacity or need for the elaborate operating systems that had been developed for mainframes and minis; minimalistic operating systems were developed. One notable early operating system was CP/M, which was supported on many early microcomputers and was largely cloned in creating MS-DOS, which became wildly popular as the operating system chosen for the IBM PC (IBM's version of it was called IBM-DOS or PC-DOS), its successors making Microsoft one of the world's most profitable companies. The major alternative throughout the 1980s in the microcomputer market was Mac OS, tied intimately to the Apple Macintosh computer.

By the 1990s, the microcomputer had evolved to the point where, as well as extensive GUI facilities, the robustness and flexibility of operating systems of larger computers became increasingly desirable. Microsoft's response to this change was the development of Windows NT, which served as the basis for Microsoft's entire operating system line starting in 1999. Apple rebuilt their operating system on top of a Unix core as Mac OS X, released in 2001. Hobbyist-developed reimplementations of Unix, assembled with the tools from the GNU project, also became popular; versions based on the Linux kernel are by far the most popular, with the BSD derived UNIXes holding a small portion of the server market.

The growing complexity of embedded devices has a growing trend to use embedded operating systems on them.

Today

Command line interface (or CLI) operating systems can operate using only the keyboard for input. Modern OS's use a mouse for input with a graphical user interface (GUI) sometimes implemented as a shell. The appropriate OS may depend on the hardware architecture, specifically the CPU, with only Linux and BSD running on almost any CPU. Windows NT has been ported to other CPUs, most notably the Alpha, but not many. Since the early 1990s the choice for personal computers has been largely limited to the Microsoft Windows family and the Unix-like family, of which Linux and Mac OS X are becoming the major choices. Mainframe computers and embedded systems use a variety of different operating systems, many with no direct connection to Windows or Unix, but typically more similar to Unix than Windows.

- Personal computers
    - IBM PC compatible - Microsoft Windows and smaller Unix-variants (like Linux and BSD)
    - Apple Macintosh - Mac OS X, Windows, Linux and BSD
- Mainframes - A number of unique OS's, sometimes Linux and other Unix variants.
- Embedded systems - a variety of dedicated OS's, and limited versions of Linux or other OS's

Unix-like

The *Unix-like* family is a diverse group of operating systems, with several major sub-categories including System V, BSD, and Linux. The name "Unix" is a trademark of The Open Group which licenses it for use to any operating system that has been shown to conform to the definitions that they have cooperatively developed. The name is commonly used to refer to the large set of operating systems which resemble the original Unix.

Unix systems run on a wide variety of machine architectures. They are used heavily as server systems in business, as well as workstations in academic and engineering environments. Free software Unix variants, such as Linux and BSD, are increasingly popular. They are used in the desktop market as well, for example Ubuntu, but mostly by hobbyists.

Some Unix variants like HP's HP-UX and IBM's AIX are designed to run only on that vendor's proprietary hardware. Others, such as Solaris, can run on both proprietary hardware and on commodity x86 PCs. Apple's Mac OS X, a microkernel BSD variant derived from NeXTSTEP, Mach, and FreeBSD, has replaced Apple's earlier (non-Unix) Mac OS. Over the past several years, free Unix systems have supplanted proprietary ones in most instances. For instance, scientific modeling and computer animation were once the province of SGI's IRIX. Today, they are dominated by Linux-based or Plan 9 clusters.

The team at Bell Labs who designed and developed Unix went on to develop Plan 9 and Inferno, which were designed for modern distributed environments. They had graphics built-in, unlike Unix counterparts that added it to the design later. Plan 9 did not become popular because, unlike many Unix distributions, it was not originally free. It has since been released under Free Software and Open Source Lucent Public License, and has an expanding community of developers. Inferno was sold to Vita Nuova and has been released under a GPL/MIT license.

Microsoft Windows

The *Microsoft Windows* family of operating systems originated as a graphical layer on top of the older MS-DOS environment for the IBM PC. Modern versions are based on the newer Windows NT core that first took shape in OS/2 and borrowed from OpenVMS. Windows runs on 32-bit and 64-bit Intel and AMD computers, although earlier versions also ran on the DEC Alpha, MIPS, and PowerPC architectures (some work was done to port it to the SPARC architecture).

As of 2004, Windows held a near-monopoly of around 90% of the worldwide desktop market share, although this is thought to be dwindling due to the increase of interest focused on open source operating systems. [1] It is also used on low-end and mid-range servers, supporting applications such as web servers and database servers. In recent years, Microsoft has spent significant marketing

and R&D money to demonstrate that Windows is capable of running any enterprise application (see the TPC article).

Other

Mainframe operating systems, such as IBM's z/OS, and embedded operating systems such as VxWorks, eCos, and Palm OS, are usually unrelated to Unix and Windows, except for Windows CE, Windows NT Embedded 4.0 and Windows XP Embedded which are descendants of Windows, and several *BSDs, and Linux distributions tailored for embedded systems. OpenVMS from Hewlett-Packard (formerly DEC), is still under active development.

Older operating systems which are still used in niche markets include the Windows-like OS/2 from IBM; Mac OS, the non-Unix precursor to Apple's Mac OS X; BeOS; RISC OS; and AmigaOS.

Research and development of new operating systems continues. GNU HURD is designed to be backwards compatible with Unix, but with enhanced functionality and a microkernel architecture. Microsoft Singularity is a research project to develop an operating system with better memory protection.

3 A history of personal computers

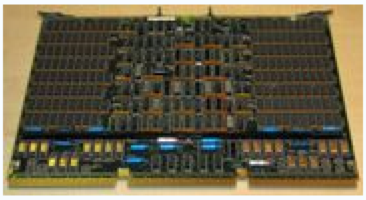
A **personal computer** (**PC**) is usually a microcomputer whose price, size, and capabilities make it suitable for personal usage. The term was popularized by IBM marketing.

Time share "terminals" to central computers were sometimes used before the advent of the PC. (A smart terminal — televideo ASCII character mode terminal made around 1982.)

Before their advent in the late 1970s to the early 1980s, the only computers one might have used if one were privileged were "computer-terminal based" architectures owned by large institutions. In these, the technology was called "computer time share systems", and used minicomputers and main frame computers. These central computer systems frequently required large rooms — roughly, a handball-court-sized room could hold two to three small minicomputers and its associated peripherals, each housed in cabinets much the size of three refrigerators side by side (with blinking lights and tape drives). In that era, mainframe computers occupied whole floors; a big hard disk was a mere 10–20 Megabytes mounted on a cabinet the size of a small chest-type freezer. Earlier PCs were generally called desktop computers, and the slower Pentium-based personal computer of the late 1990s could easily outperform the advanced minicomputers of that era.

Since the terms "personal computer" and "PC" have been introduced to vernacular language, their meanings and scope have changed somewhat. The first generations of personal microcomputers were usually sold as kits or merely instructions, and required a somewhat skilled person to assemble and operate them. These were usually called microcomputers, but personal computer was also used. Later generations were sometimes interchangeably called by the names

"home computer" and "personal computer." By the mid-1980s, "home computer" was becoming a less common label in favor of "personal computer." These computers were pre-assembled and required little to no technical knowledge to operate. In today's common usage, personal computer and PC usually indicate an IBM PC compatible. Because of this association, some manufacturers of personal computers that are not IBM PCs avoid explicitly using the terms to describe their products. Mostly, the term PC is used to describe personal computers that use Microsoft Windows operating systems.



A four-megabyte RAM card measuring about 22 by 15 inches; made for the VAX 8600 minicomputer (circa 1986). Dual in-line package (DIP) Integrated circuits populate nearly the whole board; the RAM chips are in the majority located in the rectangular areas to the left and right. One early use of "personal computer" appeared in a 3 November 1962, *New York Times* article reporting John W. Mauchly's vision of future computing as detailed at a recent meeting of the American Institute of Industrial Engineers. Mauchly stated, "There is no reason to suppose the average boy or girl cannot be master of a personal computer." [1] Some of the first computers that might be called "personal" were early minicomputers such as the LINC and PDP-8. By today's standards they were very large (about the size of a refrigerator) and cost prohibitive (typically tens of thousands of US dollars), and thus were rarely purchased by an individual. However, they were much smaller, less expensive, and generally simpler to operate than many of the mainframe computers of the time. Therefore, they were accessible for individual laboratories and research projects. Minicomputers largely freed these organizations from the batch processing and bureaucracy of a commercial or university computing center.

In addition, minicomputers were relatively interactive and soon had their own operating systems. Eventually, the minicomputer included VAX and larger minicomputers from Data General, Prime, and others. The minicomputer era largely was a precursor to personal computer usage and an intermediary step from mainframes.

Development of the single-chip microprocessor was an enormous catalyst to the popularization of cheap, easy to use, and truly personal computers. Arguably the first true "personal computer" was the Altair 8800, which brought affordable computing to an admittedly select market in the 1970s. However, it was arguably this computer that spawned the development of both Apple Computer as well as Microsoft, spawning the Altair BASIC programming language interpreter, Microsoft's first product. The first generation of microcomputers (computers based on a microprocessor) that appeared in the mid-1970s, due to the success of the Steve Wozniak-designed Apple Computer release, the Apple II, were usually known as home computers. These were less capable and in some ways less versatile than large business computers of the day. They were generally used by computer enthusiasts for learning to program, running simple office/productivity applications, electronics interfacing, and general hobbyist pursuits.

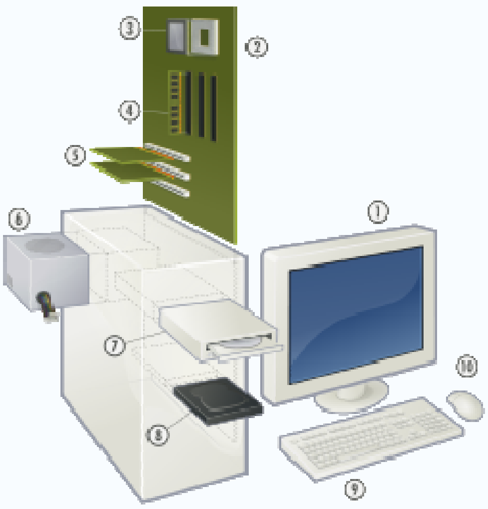
It was the launch of the VisiCalc spreadsheet, initially for the Apple II (and later for the Atari 8-bit family, Commodore PET, and IBM PC) that became the "killer app" that turned the microcomputer into a business tool. This was followed by the August 1981 release of the IBM PC which would revolutionize the computer market. The Lotus 1-2-3, a combined spreadsheet (partly based on VisiCalc), presentation graphics, and simple database application, would become the PC's own killer app. Good word processor programs would also appear for many home computers, in particular the introduction of Microsoft Word for the Apple Macintosh in 1985 (while earlier versions of Word had been created for the PC, it became popular initially through the Macintosh).

In the January 3, 1983 issue of *Time magazine* the personal computer was named the "Machine of the Year" or its Person of the Year for 1982. During the 1990s, the power of personal computers increased radically, blurring the formerly sharp distinction between personal computers and multi-user computers such as mainframes. Today higher-end computers often distinguish themselves from personal computers by greater reliability or greater ability to multitask, rather than by brute CPU ability.

Uses

Personal computers are normally operated by one user at a time to perform such general purpose tasks as word processing, internet browsing, e-mail and other digital messaging, multimedia playback, video game play, computer programming, etc. Other more specific functions usually performed with the help of a PC include working, teleworking, learning, researching, printing, online banking, online shopping and dealing online with public sector institutions and services. The user of a modern personal computer may have significant knowledge of the operating environment and application programs, but is not necessarily interested in programming nor even able to write programs for the computer. Therefore, most software written primarily for personal computers tends to be designed with simplicity of use, or "user-friendliness" in mind. However, the software industry continuously provide a wide range of new products for use in personal computers, targeted at both the expert and the non-expert user.

4 Configuration



Exploded view of a modern personal computer:

1. Display
2. Motherboard
3. CPU (Microprocessor)
4. Primary storage (RAM)
5. Expansion cards
6. Power supply
7. Optical disc drive
8. Secondary storage (HD)
9. Keyboard
10. Mouse

Personal computers can be categorized by size and portability:

- Desktop computers
- Laptop or notebooks
- Personal digital assistants (PDAs)
- Portable computers

- Tablet computers
- Wearable computers

Most personal computers are standardized to the point that purchased software is expected to run with little or no customization for the particular computer. Many PCs are also user-upgradable, especially desktop and workstation class computers. Devices such as main memory, mass storage, even the motherboard and central processing unit may be easily replaced by an end user. This upgradeability is, however, not indefinite due to rapid changes in the personal computer industry. A PC that was considered top-of-the-line five or six years prior may be impractical to upgrade due to changes in industry standards. Such a computer usually must be totally replaced once it is no longer suitable for its purpose. This upgrade and replacement cycle is partially related to new releases of the primary mass-market operating system, which tends to drive the acquisition of new hardware and tends to obsolete previously serviceable hardware (see planned obsolescence).
The hardware capabilities of personal computers can sometimes be extended by the addition of expansion cards connected via an expansion bus. Some standard peripheral buses often used for adding expansion cards in personal computers as of 2005 are PCI, AGP (a high-speed PCI bus dedicated to graphics adapters), and PCI Express. Most personal computers as of 2005 have multiple physical PCI expansion slots. Many also include an AGP bus and expansion slot or a PCI Express bus and one or more expansion slots, but few PCs contain both buses.

5 Display

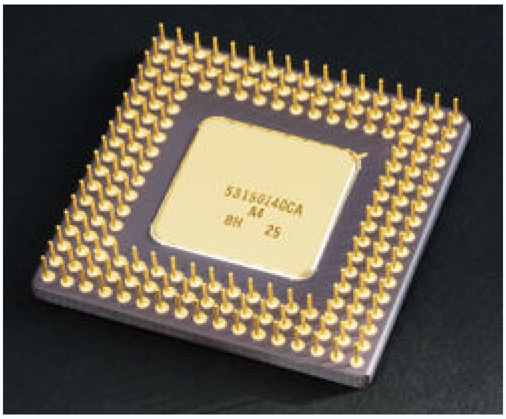
A **computer display** (also known as a **computer monitor**, **computer screen**, or **computer video display**) is a device that can display signals generated by a computer as images on a screen. There are many types of monitors, but they generally conform to display standards. Once an essential component of computer terminals, computer displays have long since become standardised peripherals in their own right.

6 Motherboard

The motherboard (or mainboard) is the primary circuit board for a personal microcomputer. Many other components connect directly or indirectly to the motherboard. Motherboards usually contain one or more CPUs, supporting circuitry and ICs for CPU operation, main memory, and facilities for initial setup of the computer immediately after being powered on (often called boot firmware or a BIOS). In many portable and embedded personal computers, the motherboard houses nearly all of the PC's core components. Often a motherboard will also contain one or more peripheral buses and physical connectors for expansion purposes. Sometimes a secondary daughter board is connected with the motherboard to provide further expandability or to satisfy space constraints.

7 Central processing unit

The central processing unit, or CPU, is the part of the computer that executes software programs, including the operating system. Nearly all PCs contain a type of CPU known as a microprocessor. The microprocessor often plugs into the motherboard using one of many different types of sockets. IBM PC compatible computers use an x86-compatible processor, usually made by Intel, AMD, VIA Technologies or Transmeta. Apple Macintosh processors were based on the Power PC (a RISC architecture) but as of 2005, Apple has used x86 compatible processors from Intel.



Intel 80486DX2 microprocessor in a ceramic PGA package

A **central processing unit** (**CPU**), or sometimes simply **processor,** is the component in a digital computer that interprets instructions and processes data contained in computer programs. CPUs provide the fundamental digital computer trait of programmability, and are one of the necessary components found in computers of any era, along with primary storage and input/output facilities. A CPU that is manufactured using integrated circuits is known as a microprocessor. Since the mid-1970s, single-chip microprocessors have almost totally replaced all other types of CPUs, and today the term "CPU" is usually applied to some type of microprocessor.

The phrase "central processing unit" is, in general terms, a description of a certain class of logic machines that can execute complex computer programs. This broad definition can easily be applied to many early computers that existed long before the term "CPU" ever came into widespread usage. However, the term itself and its initialism have been in use in the computer industry at least since the early 1960s (Weik 1961). The form, design and implementation of CPUs have changed dramatically since the earliest examples, but their fundamental operation has remained much the same.

Early CPUs were custom-designed as a part of a larger, usually one-of-a-kind, computer. However, this costly method of designing custom CPUs for a particular application has largely given way to the development of inexpensive and standardized classes of processors that are suited for one or many purposes. This standardization trend generally began in the era of discrete transistor mainframes and minicomputers and has rapidly accelerated with the popularization of the integrated circuit (IC). The IC has allowed increasingly complex CPUs to be designed and manufactured in very small spaces (on the order of millimeters). Both the miniaturization and standardization of CPUs have increased the presence of these digital devices in modern life far beyond the limited application of dedicated computing machines. Modern microprocessors appear in everything from automobiles to cell phones to children's toys.

8 Primary storage

Primary storage, or **internal memory**, is computer memory that is accessible to the central processing unit of a computer without the use of computer's input/output channels. Primary storage is used to store data that is likely to be in active use. Primary storage is typically very fast, in the case of RAM which is also volatile, losing the stored information in an event of power loss, and quite expensive. ROM is not volatile, but not suited to storage of large quantities of data because it is expensive to produce. Typically, ROM must also be completely erased before it can be rewritten, making large scale use impractical, if not impossible. Therefore, separate **secondary storage**, or external memory, is usually required for long-term persistent storage.

Confusingly, the term primary storage has recently been used in a few contexts to refer to online storage (hard disks), which is usually classified as secondary storage.

Primary storage may include several types of storage, such as main storage, cache memory, and special registers, all of which can be directly accessed by the processor. Primary storage can be accessed *randomly*, that is, accessing any location in storage at any moment takes the same amount

of time. A particular location in storage is selected by its physical memory address. That address remains the same, no matter how the particular value stored there changes.

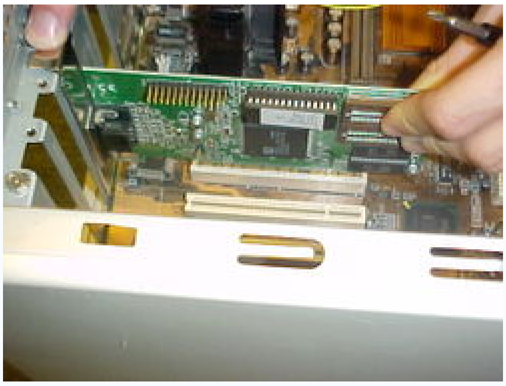
8.1 Technology and history

Today, primary storage is typically random access memory, a type of semiconductor memory. Over the history of computing hardware, a variety of technologies have been used for primary storage. Some early computers used mercury delay lines, in which a series of acoustic pulses were sent along a tube filled with mercury. When the pulse reached the end of the tube, the circuitry detected whether the pulse represented a binary 1 or 0 and caused the oscillator at the beginning of the line to repeat the pulse. Other early computers stored primary memory on rapidly rotating magnetic drums.

Modern primary storage devices include:

- Random access memory (RAM) - includes VRAM, WRAM, NVRAM
- Read-only memory (ROM)

9 Expansion card



Fitting an expansion card into a motherboard

An **expansion card** in computing is a printed circuit board that can be inserted into an expansion slot of a computer motherboard to add additional functionality to a computer system. One edge of the expansion card holds the contacts that fit exactly into the slot. They establish the electrical contact between the electronics (mostly integrated circuits) on the card and on the motherboard.

Connectors mounted on the bracket allow the connection of external devices to the card. Depending on the form factor of the motherboard and case, around one to seven expansion cards can be added to a computer system. There are also other factors involved in expansion card capacity. For example, some expansion cards need two slots like some NVidia GeForce FX graphics cards and there is often a space left to aid cooling on some high-end cards.

9.1 History of the expansion card

The first microcomputer to feature a slot-type expansion card bus was the Altair 8800, developed 1974-1975. Initially, implementations of this bus were proprietary (such as the Apple II and Macintosh), but by 1982 manufacturers of Intel 8080/Zilog Z80-based computers running CP/M had settled around the S-100 standard. IBM introduced the XT bus with the first IBM PC in 1983. XT was replaced with ISA in 1984. IBM's MCA bus, developed for the PS/2 in 1987, was a competitor to ISA, but fell out of favor due to the latter's industry-wide acceptance. EISA, the 16-bit extended version of ISA, was common on PC motherboards until 1997, when Microsoft declared it as "legacy" subsystem in the PC 97 industry white-paper. VESA Local Bus, an early expansion bus that was inherently tied to the 80486 CPU, became obsolete (along with the processor) when Intel launched the Pentium processor in 1996.

The PCI bus was introduced in 1991 as replacement for ISA. The standard (now at version 2.2) is still found on PC motherboards to this day. Intel introduced the AGP bus in 1997 as a dedicated

video acceleration solution. Though termed a bus, AGP supports only a single card at a time. Both of these technologies are now slated to be replaced by PCI-Express, beginning in 2005. This latest standard, approved in 2004, implements the logical PCI protocol over serial communication interface.

Expansion slot standards

- PCI Express
- AGP
- PCI
- ISA
- MCA
- VLB
- CardBus/PC card/PCMCIA (for notebook computers)
- Compact flash (for handheld computers)

Expansion card types

- Graphics card
- Sound card
- Network card
- TV card
- Modems
- Wireless network (such as WiFi) cards.
- Hard disk/RAID controllers (host adapter)
- POST cards
- Physics cards, only recently became commercially available

10 Power supply

A **power supply** (sometimes known as a **power supply unit** or **PSU**) is a device or system that supplies electrical or other types of energy to an output load or group of loads. The term is most commonly applied to electrical energy supplies.

The complete range of power supplies is very broad, and could be considered to include all forms of energy conversion from one form into another. Conventionally though, the term is usually confined to electrical or mechanical energy supplies. Constraints that commonly affect power supplies are the amount of power they can supply, how long they can supply it for without needing some kind of refueling or recharging, how stable their output voltage or current is under varying load conditions, and whether they provide continuous power or pulses.

The voltage regulation of power supplies is done by incorporating circuitry to tightly control the output voltage and/or current of the power supply to a specific value. The specific value is closely maintained despite variations in the load presented to the power supply's output, or any reasonable voltage variation at the power supply's input.

Electrical power supplies

A "wall wart" style variable DC power supply with its cover removed. Simpler AC supplies have nothing inside the case except the transformer.

This term covers the mains power distribution system together with any other primary or secondary sources of energy such as:

- Conversion of one form of electrical power to another desired form and voltage. This typically involves converting 120 or 240 volt AC supplied by a utility company (see electricity generation) to a well-regulated lower voltage DC for electronic devices. For examples, see switched-mode power supply, linear regulator, rectifier and inverter (electrical).
- Batteries
- Chemical fuel cells and other forms of energy storage systems
- Solar power
- Generators or alternators (particularly useful in vehicles of all shapes and sizes, where the engine has rotational power to spare, or in semi-portable units containing an internal combustion engine and a generator) (For large-scale power supplies, see electricity generation.) Low voltage, low power DC power supply units are commonly integrated with the devices they supply, such as computers and household electronics.

11 Computer power supply

A **computer power supply** typically is designed to convert 120 V or 240 V AC power from the electrical company to usable power for the internal components of the computer. The most common computer power supply is built to conform with the ATX form factor. This enables different power supplies to be interchangeable with different components inside the computer. ATX power supplies also are designed to turn on and off using a signal from the motherboard (PS-ON wire), and provide support for modern functions such as the Standby mode of many computers.

Computer power supplies are rated for certain wattages based on their maximum output power. Typical wattages range from 200 W to 500 W, although some new personal computers with high energy requirements may draw as much as 1000 W (1 kW).

Most computer power supplies have a large bundle of wires emerging from one end. One connector attached to the opposite end of some wires goes to the motherboard to provide power. The PS-ON wire is located in this connector. The connector for the motherboard is typically the largest of all the connectors. There are also other, smaller connectors, most of which have four wires: two black, one red, and one yellow. Unlike the standard electrical wire color-coding, each black wire is a Ground, the red wire is +5 V, and the yellow wire is +12 V.

Inside the computer power supply is a complex arrangement of electrical components, ranging from diodes to capacitors to transformers. Also, many power supplies have metal heatsinks and fans to dissipate large amounts of heat produced. It is dangerous to open a power supply while

it is connected to an electrical outlet as high voltages may be present even while the unit is switched off.

In desktop computers, the power supply is a small (PSU) box inside the computer; it is an important part of the computer because it provides electrical power in a form that is suitable for every other component inside or attached to the computer in order for it to work. If only a small voltage is needed, the mains power needs to be transformed to a suitable level in order for the component to work.

In portable computers there is usually an external power supply that produces low voltage DC power from a mains electrical supply (typically a standard AC wall outlet). Circuitry inside the portable computer uses this transformed power to charge the battery as needed, in addition to providing the various voltages required by the other components of the portable computer.

11.1 Domestic mains adaptors

A **power supply** (or in some cases just a transformer) that is built into the top of a plug is known as a **wall wart**, **power brick**, or just **power adapter**.

11.2 Linear power supply

A simple AC powered **linear power supply** uses a transformer to convert the voltage from the wall outlet to a lower voltage. A diode circuit (generally either a single diode or an array of diodes called a diode bridge but other configurations are possible) then rectifies the AC voltage to pulsating DC. A capacitor smooths out most of the pulsating of the rectified waveform to give a DC voltage with some ripple. Finally depending on the requirements of the load a linear regulator may be used to reduce the voltage to the desired output voltage and remove the majority of the remaining ripple. It may also provide other features such as current limiting.

11.3 Switched-mode power supply

In a **switched-mode power supply** the incoming power is passed through a transistor and transformer network that switches on and off thousands to millions of times per second. This means that a smaller, less expensive, lighter transformer can be used, because the voltage is being made to alternate faster, and thus a smaller magnetic core can be used.

Switching power supplies can be used as DC to DC converters. In this application, the power supply is designed to accept a limited range DC input and then output a different DC voltage. This is particularly useful in portable devices, as well as power distribution in large electronic equipment. A transformerless switching power supply that outputs a voltage higher than its input voltage is typically called a *boost converter*. A transformerless switching power supply that outputs a voltage lower than its input voltage is typically called a *buck converter*. These transformerless switching power supplies use an inductor as the primary circuit element in converting the voltage. Circuitry is used to pass current through the inductor to store a certain amount of electrical energy as a magnetic field. The current flow is then stopped, and the magnetic field collapses causing the stored energy to be released as current again. This is done rapidly (up to millions of times per second). By carefully metering the amount of energy stored in the inductor, the current released by the inductor can be regulated thus allowing the output voltage to be tightly regulated. A switching power supply incorporating a transformer can provide many output voltages simultaneously, and is typically called a *flyback converter*. Switching power supplies are typically very efficient if well designed, and therefore waste very little power as heat. Because of these efficiencies, they are typically much smaller and lighter than an equivalently rated linear supply.

Power conversion

The term "**power supply**" is sometimes restricted to those devices that ***convert*** **some other form of energy into electricity** (such as solar power and fuel cells and generators). A more

accurate term for devices that convert one form of electric power into another form of electric power (such as transformers and linear regulators) is **power converter**.

Uses in aviation

The most exotic power supplies are used in aviation to enable reliable restarting of stalled engines.

In jet transports, an engine is restarted from the power produced by the 400 Hz, three-phase AC generators attached to the shafts of the other engine(s). Most of the starting torque generated by the engine's motor/generator is provided by the current at the peaks of the AC waveform.

If the aircraft electronics used simple rectifying power supplies, they would use current only from these peaks, since the diodes conduct only during the voltage peaks where the input voltage is higher than the output voltage. This could prevent the pilot from restarting an engine in an emergency.

Therefore, aircraft power supplies take energy evenly from all parts of the AC waveform. this is done by using a switching power supply technique called "power factor correction" which creates a balanced current draw over the entire AC waveform.

12 Optical disc

In computing, sound reproduction, and video, an **optical disc** is flat, circular, usually polycarbonate disc whereon data is stored. This data is generally accessed when a special material on the disc (often aluminum) is illuminated with a laser diode.

David Paul Gregg developed an analog optical disk for recording video and patented it in 1961 and 1969 (U.S. patent 3430966). Of special interest is U.S. 4,893,297, first filed in 1968 and issued in 1990, so that it will be a source of royalty income for Pioneer’s DVA until 2007. It encompasses systems such as CD, DVD, and even Blu-ray Disc. Gregg's company, Gauss Electrophysics, was acquired, along with Gregg's patents, by MCA in the early 1960s.

Parallel, and probably inspired by the developments in the U.S., a small group of physicists started their first optical videodisc experiments at Philips Research in Eindhoven, The Netherlands in 1969. In 1975, Philips and MCA decided to join forces. In 1978, much too late, the long waited laserdisc was introduced in Atlanta. MCA delivered the discs and Philips the players. It turned out to be a total technical and commercial failure, and quite soon the Philips/MCA cooperation came to an end. In Japan and the U.S., Pioneer has been successful with the videodisc till the advent of DVD.

Philips and Sony formed a consortium in 1979 to develop a digital audio disc, which resulted in the very successful introduction of the compact disc in 1983.

The promotion of standardised optical storage is undertaken by the Optical Storage Technology Association (OSTA).

The information on an optical disc is stored sequentially on a continuous spiral track from the innermost track to the outermost track.

12.1 First-generation optical discs

Optical discs were initally used for storing music and software. The Laserdisc format stored analog video, but it fought an uphill battle against VHS.

- Compact disc (CD)
- Laserdisc
- Magneto-optical disc

12.2 Second-generation optical discs

These discs were invented roughly in the 1990s. Second-generation optical discs were created to store large amounts of data, including TV-quality digital video.

- Minidisc
- Digital Versatile Disc (DVD)
- Digital Multilayer Disk

- Digital Video Express
- Fluorescent Multilayer Disc
- GD-ROM
- Phase-change Dual
- Universal Media Disc

12.3 Third-generation optical discs

Major third-generation optical discs are currently in development. They will be optimal for storing high-definition video and extremely large video games.

- Blu-ray Disc
- Enhanced Versatile Disc
- Forward Versatile Disc
- Holographic Versatile Disc
- HD DVD
- Ultra Density Optical
- Professional Disc for DATA
- Versatile Multilayer Disc

13 Secondary storage

In computer storage, **secondary storage**, or **external memory**, is computer memory that is not directly accessible to the central processing unit of a computer, requiring the use of computer's input/output channels. Secondary storage is used to store data that is not in active use. Secondary storage is usually slower than **primary storage**, or internal memory, but also almost always has higher storage capacity and is non-volatile, preserving the stored information in an event of power loss.

Storage devices in this category include:

- CD, CD-R, CD-RW
- DVD
- Flash memory
- Floppy disk
- Hard disk
- Magnetic tape
- Paper tape
- Punch card
- RAM disk

14 Computer keyboard

A **computer keyboard** is a peripheral modeled after the typewriter keyboard. Keyboards are designed for the input of text and characters, and also to control the operation of the computer. Physically, computer keyboards are an arrangement of rectangular or near-rectangular buttons, or "keys". Keyboards typically have characters engraved or printed on the keys; in most cases, each press of a key corresponds to a single written symbol. However, to produce some symbols requires pressing and holding several keys simultaneously, or in sequence; other keys do not produce any symbol, but instead affect the operation of the computer, or the keyboard itself. See input method editor.

A 104-key PC US English QWERTY keyboard layout

Roughly 50% of all keyboard keys produce letters, numbers or signs (characters). Other keys can produce actions when pressed, and other actions are available by simultaneously pressing more than one action key.

15 Mouse (computing)

Fig. 1. Operating mechanical mouse

Operating a mechanical mouse (Fig. 2).
1: Moving the mouse turns the ball.
2: X and Y rollers grip the ball and transfer movement.
3: Optical encoding disks include light holes.
4: Infrared LEDs shine through the disks.
5: Sensors gather light pulses to convert to X and Y velocities.

Fig. 2. The first computer mouse

A **mouse** is a handheld pointing device for computers, being a small object fitted with one or more buttons and shaped to sit naturally under the hand. The underside of the mouse houses a device that detects the mouse's motion relative to the flat surface on which it moves. The mouse's 2D motion is typically translated into the motion of a pointer on the display.

It is called a mouse primarily because the cord on early models resembled the rodent's tail, and also because the motion of the pointer on the screen can be mouse-like (Fig. 2).

16 Main memory

A PC's main memory place (or primary storage) is fast storage space that is directly accessible by the CPU. It is generally used for storing relatively short-term data needed for software execution. Main memory is usually much faster than mass storage devices like hard disks or optical discs, but usually cannot retain data for more than a few fractions of a second without power and is more expensive. Therefore, it is not generally suitable for long-term or archival data storage. As with the CPU, most PCs use some form of semiconductor random access memory such as DRAM or SRAM as their primary storage.

17 Hard disk drive

The disk drives use a sealed head/disk assembly (HDA) which was first introduced by IBM's "Winchester" disk system. The use of a sealed assembly allowed the use of positive air pressure to drive out particles from the surface of the disk, which improves reliability.

If the mass storage controller provides for expandability, a PC may also be upgraded by the addition of extra hard disk or optical drives. For example, DVD-ROMs, CD-ROMs, and various optical disc recorders may all be added by the user to certain PCs. Standard internal storage device interfaces are ATA, Serial ATA, SCSI, and CF+ Type II in 2005.

18 Graphics - Video card

The graphics card - otherwise called a graphics adapter, video adapter, or video card - processes and renders the graphics output from the computer to the VDU or computer monitor and is an essential part of the modern computer. On older and budget models graphics cards tended to be integrated with the motherboard but, more commonly, they are supplied in PCI, AGP, or PCI Express format. Graphic cards are also the most glamorised computer component as it is the component which creates all the visual effects on the computer which is essential for playing games.

Laptop computers

A laptop computer or simply laptop (also notebook computer or notebook) is a small personal computer designed for mobility. Usually all of the peripherals needed to operate the laptop are built in to a single unit. Most laptops contain batteries to facilitate operation without a readily available electrical outlet.

Non IBM-compatible personal computers

Though many personal computers are IBM PC compatible using either Microsoft Windows or closed and open-source Unix-likes such as Linux, a number of other personal computer types are also popular.

A leading alternative to the IBM PC is the Apple Macintosh, a combination of proprietary hardware and operating system. The Macintosh orignally used the Motorola 68000 series, then shifted to the IBM and Motorola PowerPC processors.

In January 2006, Apple shifted its architecture to the same Intel chip found in IBM compatibles, allowing their computers to run Apple's own Mac OS X as well as other IBM PC Compatible Operating Systems.

Further PC and PW (Personal Workstation) types through time:

- Amiga (previously produced by Commodore, now under license from Amiga Inc.)

- Acorn Archimedes & RiscPC
- Atari ST
- BeOS BeBox
- Pegasos
- NEC PC-9800 (At one time, in Japan)
- NeXT workstations
- Sun SPARCstation
- SGI workstations like the SGI Indigo and SGI Onyx

The term "personal computer" is often avoided by advocates of the above computer systems, ostensibly because of the association it has to the "PC" in "IBM PC".

19 Disk operating system

Disk Operating System (specifically) and **disk operating system** (generically), most often abbreviated as **DOS** (not to be confused with the DOS family of disk operating systems for the IBM PC compatible platform), refer to operating system software used in most computers that provides the abstraction and management of secondary storage devices and the information on them (e.g., file systems for organizing files of all sorts). Such software is referred to as a *disk* operating system when the storage devices it manages are made of rotating platters (such as hard disks or floppy disks).

In the early days of microcomputing, memory space was often limited, so the disk operating system was an extension of the operating system. This component was only loaded if it was needed. Otherwise, disk-access would be limited to low-level operations such as reading and writing disks at the sector-level.

In some cases, the *disk operating system* component (or even the operating system) was known as *DOS*.

Sometimes, a *disk operating system* can refer to the entire operating system if it is loaded off a disk and supports the abstraction and management of disk devices. An example is DOS/360. On the PC compatible platform, an entire family of operating systems was called *DOS*.

19.1 History of DOS

In the early days of computers, there were no disk drives; delay lines, punched cards, paper tape, magnetic tape, magnetic drums, were used instead. And in the early days of microcomputers, paper tape or audio cassette tape (see Kansas City standard) or nothing were used instead. In the latter case, program and data entry was done at front panel switches directly into memory or through a computer terminal / keyboard, sometimes controlled by a ROM BASIC interpreter; when power was turned off after running the program, the information so entered vanished.

Both hard disks and floppy disk drives require software to manage rapid access to block storage of sequential and other data. When microcomputers rarely had expensive disk drives of any kind, the necessity to have software to manage such devices (ie, the 'disk's) carried much status. To have one or the other was a mark of distinction and prestige, and so was having the Disk sort of an Operating System. As prices for both disk hardware and operating system software decreased, there were many such microcomputer systems.

Mature versions of the Commodore, SWTPC, Atari and Apple home computer systems all featured a disk operating system (actually called 'DOS' in the case of the Commodore 64 (*CBM DOS*), Atari 800 (*Atari DOS*), and Apple II machines (*Apple DOS*)), as did (at the other end of the hardware spectrum, and much earlier) IBM's System/360, 370 and (later) 390 series of mainframes (e.g., DOS/360: ***D**isk **O**perating **S**ystem / 360* and DOS/VSE: ***D**isk **O**perating **S**ystem / **V**irtual **S**torage **E**xtended*). Most home computer DOS'es were stored on a floppy disk always to be booted at start-up, with the notable exception of Commodore, whose DOS resided on ROM chips in the disk drives themselves, available at power-on.

In large machines there were other disk operating systems, such as IBM's VM, DEC's RSTS / RT-11 / VMS / TOPS-10 / TWENEX, MIT's ITS / CTSS, Control Data's assorted NOS variants,

Harris's Vulcan, Bell Labs' Unix, and so on. In microcomputers, SWTPC's 6800 and 6809 machines used TSC's FLEX disk operating system, Radio Shack's TRS-80 machines used TRS-DOS, their Color Computer used OS-9, and most of the Intel 8080 based machines from IMSAI, MITS (makers of the legendary Altair 8800), Cromemco, North Star, etc used the CP/M-80 disk operating system. See list of operating systems.

Usually, a disk operating system was loaded from a disk. Only a very few comparable DOSes were stored elsewhere than floppy disks; among these exceptions were the British BBC Micro's optional Disc Filing System, DFS, offered as a kit with a disk controller chip, a ROM chip, and a handful of logic chips, to be installed inside the computer; and Commodore's CBM DOS, located in a ROM chip in each disk drive.

20 Examples of disk operating systems that were extensions to the OS

- The **DOS** operating system for the Apple Computer's Apple II family of computers. This was the primary operating system for this family from 1979 with the introduction of the floppy disk drive until 1983 with the introduction of **ProDOS**; many people continued using it long after that date. Usually it was called **Apple DOS** to distinguish it from MS-DOS.
- **Commodore DOS**, which was used by 8-bit Commodore computers. Unlike most other DOS systems, it was integrated into the disk drives, not loaded into the computer's own memory.
- **Atari DOS**, which was used by the Atari 8-bit family of computers. The Atari OS only offered low-level disk-access, so an extra layer called *DOS* was booted off of a floppy that offered higher level functions such as filesystems.
- **MSX-DOS**, for the MSX computer standard. Initial version, released in 1984, was nothing but MS-DOS 1.0 ported to Z80; but in 1988 it evolved to version 2, offering facilities such as subdirectories, memory management and environment strings. The MSX-DOS kernel resided in ROM (built-in on the disk controller) so basic file access capacity was available even without the command interpreter, by using BASIC extended commands.
- **Disc Filing System** (DFS) This was an optional component for the BBC Micro, offered as a kit with a disk controller chip, a ROM chip, and a handful of logic chips, to be installed inside the computer. See also *Advanced Disc Filing System*.
- **AMSDOS**, for the Amstrad CPC computers.
- **GDOS** and **G+DOS**, for the +D and DISCiPLE disk interfaces for the ZX Spectrum.

21 Examples of Disk Operating Systems that were the OS itself

- The **DOS/360** initial/simple operating system for the IBM System/360 family of mainframe computers (it later became DOS/VSE, and was eventually just called VSE).
- The **DOS** operating system for DEC PDP-11 minicomputers (this OS and the computers it ran on were nearly obsolete by the time PCs became common, with various descendants and other replacements).
- **DOS** for the IBM PC compatible platform

The best known family of operating systems named "DOS" is that running on IBM PCs type hardware using the Intel CPUs or their compatible cousins from other makers. Any DOS in this family is usually just referred to as *DOS*. The original was licensed to IBM by Microsoft, and marketed by them as *"PC-DOS"*. When Microsoft licenced it to other hardware manufacturers, it was called *MS-DOS*. Digital Research produced a compatible variant known as *"DR-DOS"*, which was eventually taken over (after a buyout of Digital Research) by Novell. This became *"OpenDOS"* for a while after the relevant division of Novell was sold to Caldera International, now called SCO. There is also a free version named *"FreeDOS"*.

DOS consists of an input/output system, a command processor and several utilities. The utilities are individual program files found on your DOS disk. While part of DOS, these files are not needed often enough to make it necessary or practical to keep them in the computer's RAM all the time. FORMAT.COM, the program that formats blank disks, is an example of a DOS utility. Sometimes these utilities are called external commands (as opposed to internal commands which

are included as part of the file COMMAND.COM and remain resident in memory at all times; e.g., DIR and COPY).

The command processor is also a file you see on the disk, but once read into the computer's memory, it usually resides there. Some programs provide their own command processor, and there are times when the command processor will be overwritten in memory by a program and have to be reloaded when the program stops executing.

The input/output system consists of two files and a ROM (Read Only Memory) chip. While the two files are on your disks and are loaded into memory when the computer starts, they are normally hidden from your view and not available to you for changing.

22 Input/Output System

This most primitive of the DOS systems has two parts:

- **BIOS (Basic Input/Output System).** These are the fundamental routines that control the keyboard, video display and other peripherals. The BIOS is comprised of a ROM on the computer's main circuit board and the file IBMBIO.COM (or IO.SYS), one of the two hidden files on your disk.
- **Operating System.** This is the main file-handling system for the computer. Actually, two systems exist: one for disk-based files and one for non-disk peripheral devices. They are in hidden file IBMDOS.COM (or MSDOS.SYS). (IBMBIO and IBMDOS are IBM names; MS-DOS uses IO.SYS and MSDOS.SYS.)

The two systems are necessary because non-disk peripherals demand their data as strings of characters, while disks move information in large groups, known as blocks.

Command Processor

The command processor (COMMAND.COM on your disk) performs three tasks:

- **It handles critical interrupts**...that is, COMMAND.COM takes care of all demands for attention by parts of the computer. The user typing the Control-Break program break command is an example of an interrupt.
- **It handles critical errors**...that is, COMMAND.COM takes care of problems. For example, if you leave the disk drive door open during a disk operation COMMAND.COM is responsible for the error message you will see.
- **It performs end-of-program housekeeping**...that is, COMMAND.COM takes care of making the computer's memory available for other programs and reloading parts of itself if the program wrote over them.

COMMAND.COM also places the C> prompt on the screen and interprets any command(s) you might type. In short, the command processor tells the rest of DOS what to do.

Everything that DOS interacts with has a name and the names have certain rules that have to be followed. Let's look at some of them.

Default Drive

The default drive is the first disk drive on which DOS will look for a program if no drive specification is given with the filename.

How do you know what it is? Look at the prompt. The default drive letter is part of the prompt (unless someone has changed the prompt to eliminate it).

A:\> indicates that drive A (the left or top drive in a two-drive system) is the default drive. The right (or second) drive in such a system is called drive B and the first hard disk in any system is given the letter C as its drive designation.

DOS supports many more than drives A through C. In fact, if your computer has them you can specify up to 63 drive names. (This is a "Catch 22" situation. DOS can respond to 63 drive names but converts all lower case to upper case automatically so you really can't access 63 devices.) You change drives by typing the desired default drive followed by a colon at the prompt. To change to drive C type C: as shown here:

Device Names

Character oriented devices can be addressed by DOS through their names:

- **CON:** The name for the video display and keyboard.
- **AUX: or COM1:** This is the first asynchronous communications port which usually has a modem or other serial device connected to it. The second communications port is COM2:
- **PRN or LPT1:** The first parallel printer port. PRN comes from printer and LPT is an old designator derived from line printer. A colon on PRN and all device names is optional in later DOS versions. The second parallel port is LPT2:
- **CAS1:** A holdover; this is the cassette recorder port.
- **NUL:** This is a test device. Anything sent to device NUL: goes into the bit bucket (i.e., gets thrown away).

Rules for Filenames

Like devices, disk files have to be identified so DOS can address them. These filenames have specific rules.

The basic form of a filename is:

Filename.ext

The first part of the name to the left of the period is called the root name. The root name can be from one to eight characters long and cannot be the same as a device name. The second part to the right of the period is the extension. It is optional and, if used, can be one to three characters long.

The period is used between the root name and extension and must be present if there is an extension.

The following are legal and illegal characters in a filename:

- **Legal:** A-Z 0-9 $#&@!()-{}'`_~
- **Illegal:** |<>\^+=?/[]";,* plus control characters and the space

Some other operating systems allow longer file names and there are commercial utilities which link a database of long names to your short names so you can find files by using more fully descriptive names.

[*Note*: Windows allows longer file names with the space but underneath the facade the 8.3 format is maintained.]

DOS commands are issued at the prompt C:\>. Whatever you type after that prompt that is not in the COMMAND.COM standard library is assumed to be the name of a file on the default disk and DOS will search for it under one of three names (in the order listed).

If you type C\:> FILENAME

DOS will look for: FILENAME.COM or FILENAME.EXE or FILENAME.BAT

The first is a command file (note the COM extension). The second is an execution file (EXE extension). And, the third is a batch file (a series of DOS commands in a text file which you'll learn about later in this tutorial). The first file found will be read into memory and the command processor will start the program running.

Both .COM and .EXE files execute as programs. The difference between the two relates to how memory is allocated and certain parameters in the computer are set.

23 Command Syntax

Each DOS command has a mandatory part and some have an optional part. Presented here, the mandatory parts will be shown in **bold** CAPITAL LETTERS and the optional parts in lower case.

For example,

DIR d:pathname\filename.ext /p /w

is the complete command for a disk directory. Note that only DIR is necessary.

You may note the term pathname in the above command. The pathname is the full descriptive name to any location on the disk. It includes the names of all directories (see subdirectories later in this section).

In some commands you may use wildcards. A wildcard, like the joker in a card deck, can stand for any character or group of characters.

The ? represents any single character:

FILE? = FILE1 or FILED etc.

The * represents any group of characters:

. = Any file and extension

Use caution with wildcards. They can be dangerous with commands that do things like erase files. Also, in some cases a wildcard formulation can be misleading. The combination AP*EX.COM does not mean all files that start with AP and end with EX in their root name and with a COM extension. It means all files starting with AP and having an extension of COM. The EX is meaningless as it is ignored because of the asterisk.

Disk Directory

To see a listing of what is on a disk, issue the DIRectory command.

It comes with several options (shown are the most useful, not all).

DIR d:filename.ext /p /w

DIR alone will show the complete directory. With the optional filename, DIR will try to find just that file.

- The /p option causes a pause when the screen fills.
- The /w option yields a full 80-column display of just the filenames.

There are other options for sorting the listing and displaying the contents of lower-level directories.

Now we'll see what would happen when you type DIR at the prompt.

```
A:\>dir

 Volume in drive A has no label
 Volume Serial Number is 2E44-1DE7
 Directory of A:\

MVC-001F JPG        58,143  07-17-00  1:32p
MVC-002F JPG        65,563  07-17-00  1:32p
MVC-003F JPG        62,388  07-17-00  1:33p
MVC-004F JPG        76,515  07-17-00  1:33p
MVC-005F JPG        78,443  07-17-00  1:33p
MVC-006F JPG        91,106  07-17-00  1:34p
MVC-007F JPG        84,798  07-17-00  1:34p
MVC-008F JPG        87,118  07-17-00  1:34p
MVC-009F JPG        55,371  07-17-00  1:36p
MVC-010F JPG        73,377  07-17-00  1:37p
        10 file(s)        732,822 bytes
         0 dir(s)         675,840 bytes free

A:\>
```

Note several things here.

- DIR tells you what files are on the disk, how big they are, and when they were created.
- DIR also tells how many files total are in the list, how much space those take and what free space remains.

Three Simple Commands

CLS Clears the screen and puts the cursor in the home (upper left) position.

VER Shows the DOS version number on the video display. You are shown the one-digit version and two-digit revision:

MS-DOS Version 6.00

VOL d: Displays a volume label, if one exists. The label is a name you have given to the disk when it was formatted. It is used for identification purposes. (The serial number is put on the disk by the FORMAT utility.)

Volume in drive C is HANDBOOK

Volume Serial Number is 2C35-16F9

Date and Time

These two commands show and/or set the system date and time.

Early computers relied on you to set the DOS clock during the boot process. In short order peripheral makers came out with clock cards that, with the help of a battery, kept a clock going and, with the help of a program in AUTOEXEC.BAT, loaded the time into DOS for you during boot. New computers have the clock built-in and do not require a program to load the time.

If your clock battery fails, the default values will be 1-1-80 for the date and 00:00:00.00 for time. Now and again you will see files with a create date of 1/1/80; they were created on a system where the clock has failed and DOS has used its default value.

For the DATE command you can enter the date as month/day/year with hyphens or slashes, i.e., 3/1/94 or 3-1-94 are acceptable dates.

Do not enter the day of the week, even though it shows on the screen. The computer will calculate it for you. A two digit year assumes dates between 1980 and 1999. In 2000 you will have to start putting in all four digits.

The format for DATE is:
DATE <date>

On early computers the time setting required a 24-hour clock, i.e., any time after noon had to have 12 added to it, for example 3:00 pm had to be entered as 15:00. While the TIME command will still respond to this type of time, you may not also enter 3:00p and the computer is smart enough to know you mean 15:00.
The format for TIME is:
TIME <time>

On most computers these commands will change the permanent clock settings as well as changing the date/time in DOS.
Disks straight out of the package need to be formatted, that is have tracks and sectors defined so DOS can find programs and data on the disk.
The command syntax is below (only the most useful options are shown).
FORMAT d: /s /u
where

- d: defines the disk that will be formatted
- /s puts the DOS system on disk to make it bootable
- /u specifies an unconditional format (can't unformat the disk)

An example:
Problem: Format the disk in drive A: without UNFORMAT information.
Answer: The proper command is:
C>FORMAT A: /U

```
C:\>format a: /u
Insert new diskette for drive A:
and press ENTER when ready...

Formatting 1.44M
Format complete.

Volume label (11 characters, ENTER for none)?

    1,457,664 bytes total disk space
    1,457,664 bytes available on disk

          512 bytes in each allocation unit.
        2,847 allocation units available on disk.

Volume Serial Number is 2E27-17DD

Format another (Y/N)?n


C:\>
```

In order, you are asked to confirm a disk is present for formatting, then told to what capacity the disk will be formatted (press Control-C to stop the format if this is not correct), then you are given a report on format progress. At the end you are asked for a volume label (optional) and then given a report on the success of the format in terms of the number of bytes on the disk. A serial number is assigned by DOS. It is based on system time and will likely never be the same on two individual disks.

Additional Comments

Some microcomputers have 1.2 megabyte 5.25" disk drives. There is the temptation to use 360 kilobyte disks in those drives; don't do it. The track width is smaller and if you then put the 360K disks into a 360K drive, they may not work properly. Likewise, you cannot use the high density floppy disks themselves in 360K drives. The magnetic properties of the disk are such that the 360K drives won't format them.

With the introduction of 3.5" drives, higher versions of DOS are required to correctly support the new formats. The 3.5" drives come in two sizes: 720K and 1.4MB.

Unlike the 1.2MB/360K drives disks, it is possible to format to 720K in a 1.4MB 3.5" drive. All you have to do is tell the FORMAT command the track/sector combination you need:

FORMAT A: /F:720 (this tells DOS to format the disk in drive A: to 720K)

Not all versions of DOS support higher capacity disks. For example, DOS 3.2 introduced support for 3.5-inch disks, but only at 720K format. In order to format a 3.5-inch disk at 1.44MB you will need DOS 3.3 or later.

23.1 Checking the Disk

Now and again it is useful to check the integrity of the disk directory and file allocation table (FAT). The FAT is so important to the disk that there are two copies of it on each disk.
The CHKDSK program does this for you. The basic format is:

CHKDSK d:filename.ext /f /v

Using the filename causes it to be checked for continuity (i.e., being stored on contiguous sectors on the disk for more efficient access).

- /f tells DOS to automatically fix the FAT and other problems
- /v is a verbose mode that shows progress as disk checking is taking place

Example:

```
C:\>chkdsk a:
Volume Serial Number is 2E27-17DD

    1,457,664 bytes total disk space
    1,457,664 bytes available on disk

          512 bytes in each allocation unit
        2,847 total allocation units on disk
        2,847 available allocation units on disk

      593,920 total bytes memory
      548,528 bytes free

C:\>
```

Only use the version of CHKDSK that came with your version of of DOS. Crossing versions can cause great damage to a disk.

23.2 Backing Up a Floppy Disk

Floppy disks wear out after several hundred spin hours. Well before then you should have made a copy of the disk to preserve the integrity of its contents. You can, of course **FORMAT** and then **COPY** *.* to accomplish this. There is a quicker way however:

DISKCOPY d1: d2:

If you do not give drive specifications, the utility will ask for them.

All information on the target disk will be destroyed, and DISKCOPY will format the target if it is found blank. Be careful, it's easy to destroy data by putting the disks in backwards!

Problem: Copy disk A: to B:. Issue the proper command.
Answer: C:\>DISKCOPY A: B:

23.3 Erasing Files

Files you no longer need should be deleted from your disk to make room for more current files. Use the ERASE (DELete) command for this:
ERASE d:FILENAME.ext
or
DEL d:FILENAME.ext

Be careful, typographic errors in this command can bring disaster!
You are allowed to delete all files on a disk with the wildcard * (ERASE *.*), but DOS will question you.

Recovery BEFORE writing anything else to disk is possible. An UNDELETE utility started shipping with DOS 5.0. Before that commercial utilities were available.

23.4 Renaming Files

For whatever reason, you may need to change the name of a file on your disk. (Usually this is the case when you want to change a backup file to another name in order to return it to active status.)

Use this format:
REName d:OLDNAME.ext NEWNAME.ext
Wildcards are allowed, but can cause trouble if you are not careful.
The rename command will give you an error message if NEWNAME exists.

23.5 Copying Files

The COPY command is a very powerful command within DOS. With it you can create duplicates of individual files, join several files into one, and even use your computer like a simple typewriter by "copying" from the device named CON: to the device named PRN (inefficient, but OK for short notes).

Copying one file to another (copies from filename1 to filename2):
COPY d1:FILENAME1.ext d2:filename2.ext/v
/v option verifies the copy as it takes place. This adds confidence at the price of slower operation. There are other options not shown here. Wildcards are allowed.

For example,
C:\>COPY ADDRS.LST B: Copies the single file ADDRS.LST from C: to B:.
*C:\>COPY *.* B:/V* Copies all files on C: to the disk in B: and verifies the information is it is being written.
C:\>COPY ADDRS.LST Yields an error message. Can't copy a file to itself.
C:\>COPY B:.** Copies all files from drive B: to drive C:. (If a destination is not specified, the default drive and directory is used.)

Copy can also be used to concatenate (join) several files by using the following form:

***COPY** d1:FILENAME1.ext+d2:FILENAME2.ext+... d0:filename0.ext/v*

The options are the same as the previous version of the copy command.
All specified filenames (#1, #2, etc.) will be copied and joined into filename0. If no filename0 is specified, the first source file named will be used.

Wildcards are dangerous with this command.

Example:
Contents of FILE1: This is file number one
Contents of FILE2: This is file number two
C:\>COPY FILE1+FILE2 FILE3

Contents of FILE3: This is file number oneThis is file number two

The COPY command can be used to create text files by copying from device CON: to a file. The procedure is outlined in the text of the example below.
C:\>COPY CON: TEXTFILE

This is the text to go into the text file being created. Each line is typed to the screen and it is being saved into a buffer for later transfer to the file TEXTFILE. Each line may be corrected as it is typed, but cannot be changed after it is terminated by the carriage return. Also, if you happen to type beyond column 80 on the screen, you cannot correct anything on the line above. Each line must be terminated by a carriage return (the enter key). You signal you are finished by typing a Control-Z, the symbol for end-of-file, followed by Return. ^Z
1 File(s) copied

XCOPY

For copying multiple files the XCOPY command can be a powerful ally. As its name implies, the command performs extended copies.

Its format (with only often-used options) is shown here:

XCOPY d1:PATH1 d2:path2 /a /m /s /v

Like the COPY command, XCOPY can take a single drive/path designator in which case files from that destination will be copied into the current directory. Some options:
/A Copy only files with archive bit set; do not reset archive bit.
/M Copy only files with archive bit set; reset archive bit.
/S Copy subdirectories as well unless they are empty.
/V Verify copied files as they are written.

You can copy an entire hard disk to another disk with a single command:
XCOPY C: D: /S

The contents of drive C: will be copied to drive D: a file at a time, with the subdirectory structure intact.

You can use the same technique to back up a hard disk to a removable disk (e.g., Bernoulli or other removable media - don't use floppies). Note the /M option above. When DOS writes a file to the disk it sets an archive bit in the disk directory to indicate the file has been somehow changed (it's possible to write a file and not change it but DOS just assumes changes were made). The /M option for XCOPY can take advantage of this.

To proceed:
1) Make a full backup first.
Use the ~ATTRIB~ command to set all archive bits to ON:
*C:\>ATTRIB +A *.* /S*
Use XCOPY to copy all files and directories, turning all archive bits OFF in the process (assumes removable media is G:):
C:\>XCOPY C:\ G: /M /S
2) On a regular basis use the XCOPY command to perform an incremental backup:
C:\>XCOPY C:\ G: /M /S

The backup on drive G: will be an image of the file and directory structure on drive C:. The incremental backup makes certain the image is current.

Periodically, in order to purge deleted files from the backup you should start over at #1 above and a clean backup disk.

Typing a File

Any text file saved in ASCII character format can be easily seen on your video display. Use the type command:

TYPE *d:FILENAME.ext*

All characters in the file will be displayed on the screen, including any control characters, sometimes resulting in interesting displays.

Any control-I characters found will be interpreted as a tab, and spaces will be added to get the cursor over to the next 8-character boundary; some output may appear as tables. Control-Z will cause output to stop.

Attempting to TYPE a .COM or .EXE file will result in garbage on the screen, and should generally be avoided.

24 Subdirectory Introduction

As the name implies, a pathname is nothing more than a "path" that directs DOS to your particular file.

You see, with DOS 2.x, IBM/Microsoft introduced multiple directories on a single disk. In effect, this lets you sort your files into groups and place each related group into its own directory. This means you don't have to search an entire disk to find one file.

A lower-level directory is called a subdirectory (what else?)

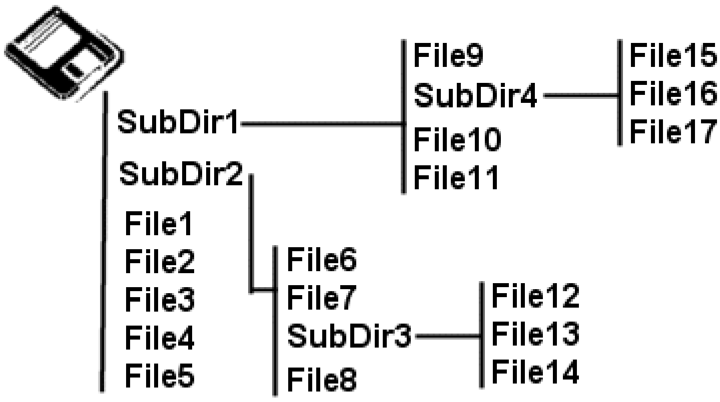
Seriously, consider a disk. To this point you have learned that each file on that disk is represented as an entry in the directory, put there so both you and DOS can find the file on disk.

If, instead of data, you created a file that pointed to other files on the disk, you will have built what amounts to a subdirectory.

DOS manipulates files in subdirectories through several directory commands and what is called a pathname.

In this section we'll look at the DOS commands for manipulating subdirectories and how we can set an environment variable (PATH) to allow DOS to find programs.

The DOS directory structure can be thought of as a tree, with the master disk directory being called the **root** and subdirectories thought of as branches. The root is the hard disk's master directory. It may contain up to 512 entries. Subdirectories may contain any number of entries (until the disk is full). A floppy root directory may contain 112 or 224 entries. A typical tree might look like...



In the example there are five files and two subdirectories in the root. Each of the subdirectories has similar contents. SubDir1, for example, has three files and one subdirectory in it. This structure can be extended until the disk is completely full, subject only to the constraint of 63 characters for the pathname that you will use to find a particular file.

The rules for a subdirectory name are just like that for filenames (eight characters followed by a period and three character extension).

They show up in a directory listing with the designator <DIR> behind them.

Let's see now how to build a pathname.

Pathnames

Assume the subdirectory structure (only directories are shown, not files)...

Root	Level 1 Subdirectory	Level 2 Subdirectory
C:\	WORDPROC	BOOK MEMOS LETTERS

This series of subdirectories was set up to categorize various files developed by a word processor. Let's move in the structure:

- \WORDPROC\LETTERS Would be the pathname from the root to subdirectory LETTERS.
- \WORDPROC Is the pathname from root to WORDPROC.

Note that each subdirectory in the path is separated by a backslash (\). The single backslash at the beginning of the pathname indicates the root. All pathnames must originate in either the current directory or root.

A test -- What is the pathname from the root to the subdirectories listed below?

- Subdirectory LETTERS
- Subdirectory BOOK
- Subdirectory WORDPROC

Answers...

- \WORDPROC\LETTERS
- \WORDPROC\BOOK
- \WORDPROC

When DOS is booted, the root directory is automatically selected. To type a file named MYMEMO.TXT in subdirectory MEMOS the command would be:

C:\> TYPE \WORDPROC\MEMOS\MYMEMO.TXT

\WORDPROC\MEMOS\ is the pathname that DOS would use to find the file MYMEMO.TXT and then show it on the screen.

If you've got work to do with files in the MEMOS subdirectory, typing the complete pathname all the time would be inefficient. Therefore, DOS gives you a method of making DOS recognize the MEMOS directory as the default: the **Change Directory** command.

To change to the MEMOS subdirectory from the root the command would be:

C:\> **CD** \WORDPROC\MEMOS

If set properly (see just below) the prompt might also change to reflect the change directory (C:\WORDPROC\MEMOS>, and a DIR command would now show the contents of the MEMOS subdirectory instead of the root and DOS would look for all command files in that subdirectory instead of the root.

An easy way to keep track of where you are in the directory tree is to use the **PROMPT** command to set a prompt that shows the current directory along with the current drive. You can easily do this by adding the line:

PROMPT PG

to your AUTOEXEC.BAT file.

Make and Remove Subdirectories

If you are going to have subdirectories, there must be a way to make them. The syntax for the make directory command is:

MD d:pathname**DIRNAME**

You can make a subdirectory IN any directory FROM any directory so long as you give the appropriate pathname. Usually, you will change to the directory you want the subdirectory to be in

and then issue a simple MD DIRNAME command. That way there is no mistake about what will happen.

When you no longer need a directory you may remove it from the disk. The first thing you have to do is empty it of files and move out of it. Only then will you be able to remove it. The syntax for removing is:

RD d:pathname**DIRNAME**

You cannot remove the root directory (it's the master for the disk and when it's the only directory you would have to be in it, and you can't remove a directory you are in).

Note: In DOS 6.0 the command **DELTREE** was introduced. DELTREE will both remove files from a directory and remove that directory with a single command.

Mysterious Dots

When you are in a subdirectory and issue the DIR command, you will see something like this:

```
A:\TUTORIAL>dir

 Volume in drive A has no label
 Volume Serial Number is 2E27-17DD
 Directory of A:\TUTORIAL

.              <DIR>        01-03-01  9:36p
..             <DIR>        01-03-01  9:36p
FILE1    TXT             5  01-03-01  9:36p
FILE2    TXT             5  01-03-01  9:36p
         2 file(s)             10 bytes
         2 dir(s)      1,456,128 bytes free

A:\TUTORIAL>
```

The dots indicate you are in a subdirectory. The single dot is the current directory and the double dots are the parent to the current directory. Thus you could move to the parent of TUTORIAL (in the example above) by simply issuing the command CD ..

Used with care, the dots can speed up subdirectory commands.

Tree

All directory paths and their relationships are called a tree. If you don't remember the various subdirectories (and optionally the files in them) DOS offers you a chance to see them with the TREE command:

TREE d:/f

This command lists all paths from the root on the disk. If you use the /f option, you will also see all files in each subdirectory. (Only the filenames are shown, not their size or creation date/time.)

For a permanent record, press Control-PrtScr before issuing the TREE command and again after. Your printer will record all text scrolling past on the screen. (Or redirect to a file with TREE d:/f > Filename.)

[*Note: TREE has been removed under Windows.*]

25 Subdirectory Review

Assume the shown subdirectory structure...

Root	Level 1 Subdirectory	Level 2 Subdirectory
C:\	WORDPROC	BOOK MEMOS LETTERS

The following series of commands perform the indicated functions. Study them and make certain you understand them.

From the root:

- MD \WORDPROC\LETTERS\HOME Makes a new directory called HOME in LETTERS.
- RD \WORDPROC\BOOK Removes BOOK from WORDPROC.
- CD \WORDPROC\MEMOS Makes MEMOS the active directory.

To move from the root to the subdirectory MEMOS type:

- CD \WORDPROC\MEMOS

Remember, to make any directory the current directory, simply issue the CD command using the pathname that you would use to access any file in that subdirectory. CD \ alone will move you back to the root from anywhere.

From the root, the following command removes the subdirectory LETTERS:

- RD \WORDPROC\LETTERS

Remember, before you can remove a subdirectory, it must be empty (all files deleted and other subdirectories removed), you cannot be in it, and you cannot remove the root directory. In DOS 6.0 you can do both with the single command **DELTREE**.

A final reminder:

Use subdirectories as needed, but don't overdo it. Pathnames are limited to 63 characters; and, you can get lost in the directory structure if you create a very complex one. **PROMPT** PG will help you keep track.

Also, directories are not free. Each takes up some disk space so if you fill your disk with directories, there won't be room for anything else.

26 BATCH FILES

A batch file is nothing more than a collection of DOS commands placed into an ASCII text file. When DOS executes a batch file, each line in the file is treated as a DOS command and is executed as if you had typed the command at the system prompt. Their main use is to automate DOS command sequences you repeat often or are hard to remember.

Each line in the file is treated as a DOS command and is executed as if you had typed the command at the system prompt.

In addition to standard DOS commands, batch files also have their own subset of commands which allow you to actually write a batch file as a small program. Branching and iteration are allowed in these programs.

Also, batch commands may have external parameters passed to them at the time you execute the file.

A batch file must have the extension .BAT and you execute the commands in it like any other DOS command: by typing the file's root name, without the extension, at the system prompt.
Parameters are extra pieces of information that you type after many of the DOS commands. For example, "DIR B: /W" contains the parameters B: and /W. These modify the basic operation of the command, but are not required by the command.

You pass parameters to a batch file in the same manner; by typing the information after the batch command, but before tapping the Enter key.

The parameters may be used in any place in the batch file where a parameter would normally be used as part of the DOS command being run.

Markers are used within the batch file to signify which parameter goes where. Markers are comprised of a percent sign (%) and a single digit between 0 and 9 (that's ten markers in use at any one time; remember, zero is a number).

As an example:

Assume that a batch file named REPEAT.BAT is on the current drive. This file contains two commands: "ECHO OFF" which stops DOS from showing commands and "ECHO %1 %3" where the ECHO command can show messages on the screen. If, at the DOS prompt you typed "REPEAT Red Blue Green" your screen would show "Red Green" as shown in the screen capture:

In this simple example, three parameters were passed to the batch file and were placed into the ECHO command in the order received. Only the first and third are shown as only those were referenced in the batch file.

The parameters and markers were related as follows...

- %1 is marker 1 and, in this example, represents "Red", the first parameter;
- %3 is marker 3 and, in this example, represents "Green", the third parameter;
- %2 would be marker 2 but, in this example, is not present so "Blue", the second parameter, is ignored.

Note: Marker zero is assigned the name of the batch file in the form you typed it in (i.e., all caps, all lower case, or a mixture).

In addition to the normal DOS commands, batch files have their own subcommand structure. Following are the subcommands in the order we will discuss them:

- **ECHO:** Turns command display on/off or may display a message.
- **REM:** Displays a message on the screen.
- **PAUSE:** Temporarily stops the batch file execution.
- **GOTO:** Jumps to a labeled set of commands in the batch file.
- **IF:** Permits conditional operation of any command.
- **SHIFT:** Reassigns the relationship of parameters to markers.
- **FOR..IN..DO:** Allows iteration subject to defined conditions.
- **CALL:** Runs another batch file then returns to first.
- **@:** Turns display off for single command

ECHO

You have seen one facet of ECHO already: it can send a message to the screen. More importantly, it can help clear the clutter on the screen when a batch file executes. Normally, the batch file commands would show on the screen as if you were typing them. This can get distracting. If you want to suppress the command echoing type:

ECHO OFF

To restart echoing, type: ECHO ON

When ECHO is off no part of the actual DOS commands in the batch file will show on the screen. To display a message put it after the ECHO command:

ECHO Message

As you will see, the REMark command also displays messages; but NOT if ECHO is off! Use ECHO if you want the message to show no mater what.

REMark

REMark can be used to send messages to the screen or simply to document some part of your batch file's operation.

Use them extensively in long batch files. The computer operator (it won't always be you) wants to know what is happening and, over time, you might forget what a complicated set of commands really does.

The format is:

REM Message

REMarks can be up to 123 characters long, although you usually won't want to go over the screen width so you can control the display.

An undocumented, variation of this command is the period. In DOS 2.x, if a line in a batch file starts with one or more periods (.) it is treated as a remark. It is dangerous to use this however since in DOS 3.x and later that same line would be treated as the path to a command!

PAUSE

The last "simple" command is **PAUSE**. Its basic function is to stop the execution of the batch file until you press a key. This can allow you to perform a necessary task; like perhaps changing a disk or verifying that a particular disk configuration is in place in order to avoid errors as the remaining parts of the batch file are executed.

In early DOS versions PAUSE would optionally display a message. It does not now. In order to display a message you have to couple PAUSE with the ECHO command. The format in use now would require:

ECHO Message
PAUSE

The message will show, followed by the DOS message:

Strike a key when ready... (in early DOS versions)
Press any key to continue... (in later DOS versions)

GOTO

The format for this command is:

GOTO LABEL

where LABEL is a line in your batch file that starts with a colon (:) followed by up to an eight character name (actually, you can type any number of characters, but DOS only recognizes the first eight).

Want to create an endless loop? GOTO gives you an easy way. Here is one example:

:START
REM...This is being typed by an endless loop.
GOTO START

When executed, this file will continue to print the endless loop line and the GOTO command until you tap Control-Break.

Use this subcommand to transfer control in your batch files.

If you want to terminate any batch file before it is complete, issue the break command [Control-Break].

Interim Review

You have now seen the simple stuff...Let's see how much of it stuck. Following are a few true/false questions. Think of the answer then look below to see if you were right.

Questions:

1. The file AUTOEXEC.BAT must be in the root directory of the boot disk.
2. Batch files can execute a variety of subcommands in addition to the standard DOS commands.
3. Parameters are designated as %0 through %9.
4. When DOS finds a batch subcommand that is improperly phrased, it stops execution of the file and shows Syntax error.
5. You may interrupt a batch command by pressing Control-Home.
6. When you type Control-Break to stop a batch file, DOS will ask you if you want to terminate. If you say No the current command will be ignored but the rest will be processed.
7. The batch filename is substituted for marker %0.8. REMark subcommands display a message regardless of the the condition of ECHO.
9. PAUSE causes the batch file to temporarily stop and wait for you to press a key.

Answers:

1. True, AUTOEXEC.BAT executes automatically when DOS boots so it must be in the root. (A trick question as it assumes knowledge from prior tutorials.)
2. True, you've seen some, with more to come.
3. False, those are markers. A parameter is information you type in on the command line that is substituted for a marker when commands are executed.
4. True, we didn't talk about that specifically, but that's what happens.
5. False, the correct command is Control-Break.
6. True, revealing another little quirk of batch file processing.
7. True, but there is a command you can use to change that as we'll see later.
8. False, REM only shows its message if ECHO is ON. If ECHO is OFF, only ECHO puts a message to the screen.
9. True.

That's enough. You should now understand the batch file subcommands you are likely to need most often.

Let's move on to the rest of the subcommands that allow you to program within a batch file. Subcommands in this section are generally used to create batch programs. As such, they are a bit more complex than the simply ones studied on the last page.

IF

The IF subcommand allows you to make decisions within your batch file. Its syntax is:

IF Condition Command where,
Command = Any legal DOS command or batch file subcommand
Condition = One of three tests that yield true or false:

1. The ERRORLEVEL of a program.
2. Two strings are equivalent.
3. A file exists in the current directory.

Unlike programming languages, which allow many logical tests in an IF statement, the batch IF statement is limited to only the three above.

Condition 1: ERRORLEVEL is a number that indicates to DOS whether the last program run was successful.

A zero (0) indicates a good run, anything above zero indicates an error condition. (Only DOS commands BACKUP and RESTORE have an exit code.)

You can create small programs to capture the keyboard output and report it as an error level (see Batch Sample #3 later).

Condition 2: String comparison is indicated by double equal signs:
String1 == String2
compares the two strings.

This comparison is often used with parameters and markers to check for a particular entry. For example,
IF %1 == 40 MODE C40
checks parameter one for 40 and, if found, changes the display to 40-columns wide.

Condition 3: The logical test checking for a file has the format:
EXIST d:Filename

You can use this test to check and see if a DOS disk is in the active drive (as one example). Another use might be to check for a particular file to see if the user has a valid disk in the drive. Pathnames are NOT allowed.

Password Example

The following batch file can be used to establish a password for running a program. The batch file is named START.BAT and calls the program named WP.COM.

```
ECHO OFF
IF %1==XYZ GOTO GOOD
ECHO BAD PASSWORD...ENDING
GOTO END
:GOOD
ECHO YOU'RE OK...STARTING
WP
:END
```

Below you'll see the response of the computer to various commands...
First the bad password. At the prompt type START ABC.

```
A>START ABC
A>ECHO OFF
BAD PASSWORD...ENDING
A>_
```

Now use the correct password. Type the correct command at the prompt.

```
A>START XYZ
A>ECHO OFF
YOU'RE OK...STARTING
```

At this point the WP program starts. You don't see the command because echo is off.

SHIFT

The limit of 10 parameters for a batch file can be raised through use of the SHIFT command. The syntax is the single word:

When that subcommand is encountered, all parameter/marker pairings are shifted one to the left. Whatever was assigned to %0 is lost, the contents of %1 are moved to %0, %2 moves to %1 ... %9 moves to %8 and a new parameter from the command line is moved into %9.
While this brings in a new parameter, all have shifted and you must anticipate the effect on your batch file "program."

The effect of SHIFT:

%0←%1←%2←%3←%4←%5←%6←%7←%8←%9
↓
Lost

Remember, this command seems very simple, but its effects are far ranging and the logical consequence of mismatching parameters and markers and be disastrous!

FOR..IN..DO

This command is similar to the programmer's FOR..NEXT loop in that a particular action can be repeated a given number of times. The syntax (which may look a bit complicated) is:

FOR %%Variable IN (Set) DO Command

%%Variable is one-letter with the mandatory %% before it. Two percentage signs are used so this variable won't be confused with a marker.

(Set) is one or more filenames or commands you want %%Variable to assume while the command is being executed. Use a space between entries, and pathnames are allowed. Don't forget the parenthesis around the set. Wildcards may be used within (Set) if you are using filenames.
Command is the particular DOS command or batch subcommand you want to have performed. Usually one or more of these commands will contain the %%Variable in it. If (Set) contains DOS commands, only %%Variable is used.

In effect, what this subcommand does is cause %%Variable to be an index into the (Set) for use by Command.

An example:
Problem: Compare all files on the disk in drive A: with those on the disk in drive B: and report those that match.
Answer: The following two-line batch file run from the A: drive will do that task:

ECHO OFF
FOR %%Z IN (.*) DO IF EXIST B:%%Z ECHO %%Z is on A: and B:*

Let's see how...

The first line turns command display off to clear the clutter.

The second line is executed as many times as there are files on the disk in A: [the set (*.*) assures this]. Each of those filenames are assigned to %%Z in turn and then checked for presence on drive B: with the EXIST logical statement. If EXIST is true, then the message at the end of the IF subcommand is sent to the screen, otherwise nothing is printed and the next file on drive A: is assigned and checked.

The following will diagram this process...
Files on drive A:

COMMAND.COM
FILE.ONE
FILE.TWO

Files on drive B:

COMMAND.COM
FILE.ONE
FILE.LTR

The batch subcommand we are investigating is:

FOR %%Z IN (.*) DO IF EXIST B:%%Z ECHO %%Z is on A: and B:*

Each filename on A: is substituted in the IF subcommand and then executed. To get the same effect you would have to type:

IF EXIST B:COMMAND.COM ECHO COMMAND.COM is on A: and B:
IF EXIST B:FILE.ONE ECHO FILE.ONE is on A: and B:
IF EXIST B:FILE.TWO ECHO FILE.TWO is on A: and B:

In the case of the example above, the first two would have a positive response and the last would not print anything. Study it carefully before going on!

Another example:
Files on drive A:

COMMAND.COM
FILE.ONE
FILE.TWO
Files on drive B:
COMMAND.COM
FILE.ONE
FILE.LTR

OK, told you to study the example. Let's see if you remember. What is the one line batch file command to find and report out all files starting with an "F" on drive A:?

FOR %%Z IN (A:F.*) DO ECHO File %%Z is on drive A:*

In this case you see that the A: disk is checked for any file starting with the letter "F" and that name is substituted in the variable %%Z. The appropriate message is then printed.

This is not an easy concept. It will require some study and practice to master.

Let's now look at using DOS commands in (Set).

The (Set) can contain DOS commands instead of filenames and these commands will then be executed in sequence (to include running large programs under the control of the FOR..IN..DO loop).

Let's say you want to sequentially:

- Clear the screen,
- Show the DOS version number,
- then Show a disk directory with pause on.

You could do all that in a one line batch file with the following command in it:

FOR %%T IN (CLS Ver Dir/P) DO %%T

When using DOS commands in (Set) you must use the space as a delimiter and cannot have the asterisk (*) or question mark (?) in any command. Use a colon (:) instead of a space when passing parameters to programs (i.e., DBASE:FILE instead of DBASE FILE). [The colon trick does not always work; you have to experiment.]

It is possible to issue the FOR..IN..DO command at the DOS prompt by dropping one of the percentage signs (%) on the variable.

CALL

Sometimes it's handy to be able to run one batch file from another. Until DOS 3.3 you had to resort to tricks to do that; without the tricks when you called a second batch file from a first then control would transfer to the second and you'd never get back to the first.

The command: **CALL** d:path**FILENAME** parameters
can be used in DOS versions 3.3 and later to start a second batch file from a parent and then return to the parent after the second finishes. *Note:* Do not use pipes or redirection on the second file.

The FILENAME is the name of the second batch file; the parameters are any options that the second file requires to properly run. When the second batch file terminates, control is returned to the first batch file on the line following the CALL command.

You can simulate CALL in DOS versions lower than 3.3 by using:

COMMAND /C d:path\FILENAME parameters

@

In DOS 3.3 and later you can selectively cause lines in a batch file from being displayed. To cause this, place an "**@**" sign in front of the line in question.

One use of this would be to suppress showing the "ECHO OFF" command which is the starting command of many batch files. Until the ECHO OFF command is actually executed, ECHO is ON and the command shows. To stop even this from showing make the first command: @ECHO OFF.

If you need to use the "@" feature with a command that already starts with @ (e.g., @WIP.EXE) then use a double @ (e.g., @@WIP) in the batch file (this should be rare).
Below are some simple batch file examples to whet your appetite.

26.1 AUTOEXEC.BAT

AUTOEXEC.BAT is a special batch file name that, if found in the root directory of the boot disk, will automatically run before control of the computer is turned over to you.

You might want to always load particular files on starting the computer and the commands to do this would be in AUTOEXEC.BAT.

Typically, the AUTOEXEC.BAT file is used to set the system up to your particular needs. This includes setting the PATH to define where DOS will look for commands, defining various variables in the DOS environment and setting the PROMPT to look like you want it to.

If you want to terminate the AUTOEXEC.BAT file (or any other batch file) before it is complete, issue the break command [Control-Break].

```
@Echo OFF
Path C:\DOS;C:\;C:\BAT;C:\UTILITY;
Prompt $p$g
Set TEMP=C:\Temp
C:\Utility\NumLock –
CD\
CLS
Type C:\Bat\Menu.TXT
```

This file sets the PATH, defines a prompt and a temporary directory, runs a utility program and then clears the screen and types a menu.

Batch Sample #1

Here is a sample batch file you might use to periodically back up and clear a word processing data disk. It assumes that word processing backup files are named with extension .BAK, you don't want them, and they are on the disk in drive B:. The batch file is called BAK.BAT and is also on the word processing data disk in drive B:. Start it by typing B:BAK.

@ECHO OFF
CLS
*ECHO * * * Word processing cleanup * * **
ECHO Put backup disk in drive A:
PAUSE
ERASE B:.BAK*
COPY B:.* A:*
DEL A:BAK.BAT

ECHO Backup is complete. Label your disk.

The first line turns screen echo off. The next line clears the screen. A label is then displayed with instructions on where to place the backup disk. A keypress needed to continue, then all .BAK files are erased and all files on drive B: are copied to A:. The batch file is then deleted from A: and a termination message shown.

Your situation might be different and you may not want to delete .BAK files first, but this should give you an idea of what you can do.

Batch Sample #2

Suppose you have a phone list in a file named FONE on your disk. There is a DOS utility called FIND which allows you to search files for specific text strings. The general format for using FIND is:

FIND "text" filename.

Instead of typing a complicated command each time, create a batch file called LOOKUP.BAT as follows:

@ECHO OFF
FIND "%1" FONE

This file is called up by typing LOOKUP NAME at the system prompt. Assuming that you have typed the name in all caps, you might see a response like:

C> LOOKUP JONES
JONES Tom (805) 555-1212 Tom owes me $50

Clever use of DOS will save you money!

Batch Sample #3

Call this file MENU.BAT. It demonstrates branching. See below for creation of GETKEY which stops the computer, waits for a keystroke and returns its ASCII value as a program error level. (The ASCII value of 1 is 49, 2 is 50 and 3 is 51.)

An IF ERRORLEVEL test is true if the error level tested is less than or equal to the actual error level. Therefore, such tests have to be in reverse order or the first will always be true.

@ECHO OFF
:START
CLS
ECHO --------------------
ECHO 1 – Wordprocessing
ECHO 2 – Spreadsheet
ECHO 3 - Exit menu to DOS
ECHO --------------------
:QUERY
ECHO 1, 2, or 3?
GETKEY
IF ERRORLEVEL 52 GOTO QUERY
IF ERRORLEVEL 51 GOTO L3
IF ERRORLEVEL 50 GOTO L2
IF ERRORLEVEL 49 GOTO L1
GOTO QUERY
:L1
Commands necessary to call up word processor
GOTO START
:L2
Commands necessary to call up spreadsheet
GOTO START
:L3
CLS

GETKEY.COM

GETKEY.COM is a small file that you need to create to allow the example above to work. If you are not comfortable with the procedure outlined below **DO NOT DO IT**. The program DEBUG can be dangerous when used improperly.
DEBUG comes with most versions of DOS. You type everything highlighted:

C:\> ***DEBUG***
*-**E 100 B4 00 CD 16 B4 4C CD 21***
*-**N GETKEY.COM***
*-**R CX***
CX 0000
*:**8***
*-**W***

Writing 0008 bytes
-Q

The DEBUG command starts the DEBUG program. The prompt for that program is a hyphen. The "E" command enters data and the "N" command names a file. The "R" command finds out what's in register CX and then enters a new value (the number of commands in this case). The "W" command tells DEBUG to write the file and "Q" quits the DEBUG program and drops you back to the DOS prompt.

You should now have the file GETKEY.COM on the default drive. Try it in a batch file like the one above.

The limits to the uses of batch files are found only in your imagination.

One batch file can call another, but the original file will loose control of the computer, so don't expect the first batch file to do anything of use after a second is called. DOS 3.3 introduced the CALL command to get around this limitation.
Some of the commands will even work directly at the DOS prompt. A batch file might not be necessary. The FOR..IN..DO command, for example, can be used outside of a batch file if you use only one percentage sign (%) to designate the variable instead of two.
If a batch file contains a syntax error in any of its commands, the file will stop execution at that point and you will be returned to the DOS prompt.
Finally, DOS "remembers" the disk that contains the batch file and the drive it was in. If you removed the original disk, DOS will ask you to replace it before going on.

A summary:

- CALL Filename
- Calls a second batch file from within a parent.
- ECHO ON -or- ECHO OFF -or- ECHO <Message>
- Turns command display on or off, or displays a message to the screen.
- FOR %%Variable IN (Set) DO Command
- Allows use of a single batch command for several different variables.
- GOTO Label
- Jumps to locations within a batch file.
- IF Condition Command
- Permits conditional execution of commands, and
- where Condition is
    - ERRORLEVEL #
    - String1 == String2
    - EXIST Filename
- PAUSE Message
- Halts execution until the user presses a key.
- REM Message
- Displays a message to the screen if ECHO is on.
- SHIFT
  Shifts command line parameters to the left by one.
- @
  Turns display off for a single command

27 Data structure in disk.

27.1 Disk Structure and Partitions

A hard disk is physically composed of a series of flat, magnetically coated platters stacked on a spindle. The spindle turns while the heads move between the platters, in tandem, radially reading/writing data onto the platters.

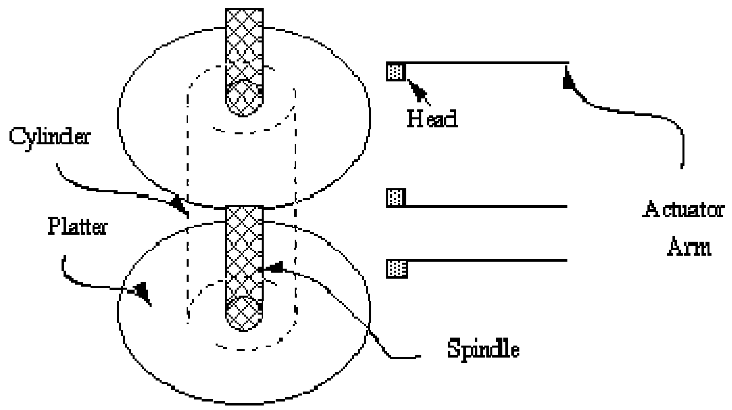


Fig. 3. Physical Disk Structure

27.2 Disk tracks, cylinders, and sectors

A disk is divided into **tracks**, **cylinders**, and **sectors**. A **track** is that portion of a disk which passes under a single stationary head during a disk rotation, a ring 1 bit wide. A **cylinder** is comprised of the set of tracks described by all the heads (on separate platters) at a single seek position. Each cylinder is equidistant from the center of the disk. A track is divided into segments of **sectors**, which is the basic unit of storage.

On Sun systems a **sector** is 512 bytes (1 disk block) of data, with header and trailer information. The latter make it possible for the controller to identify sectors, detect data errors, and perform error corrections when necessary. The actual layout of a disk sector will vary depending on the controller, but should look something like that shown in Fig. 2.3. There are two Preambles and a Postamble (whose sizes may vary due to rotational speed, etc., and are disk dependent). The Header field lets the controller know where the head is positioned, and the ECC field is for error correction.

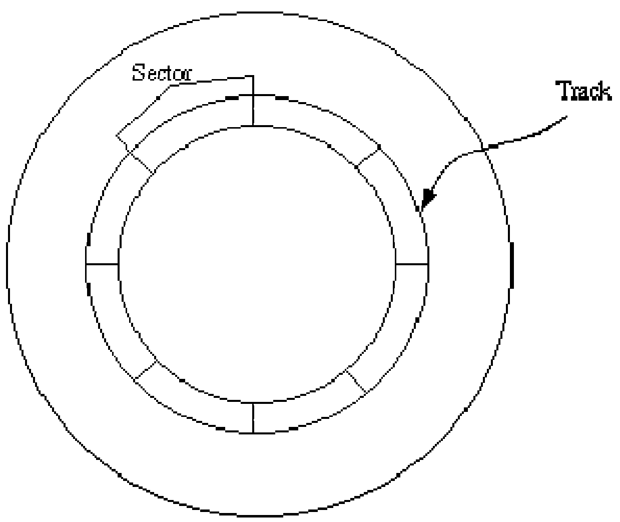


Fig. 2. Disk Platter

Preamble 1	Header	Sync	Preamble 2	Sync	Data Field	ECC	Postamble
25 bytes	8 bytes	1 byte	25 bytes	1 byte	512 bytes	6 bytes	22 bytes

Fig. 4. Sector

The number of sectors per track varies with the radius of the track on the platter. The outermost tracks is larger and can hold more sectors than the inner ones. These outer tracks also spin faster under the head than do the inner ones, because while the angular speed remains the same, the larger circumference results in more sectors spinning by in the same period for the outer tracks. Disk blocks are numbered starting at the outermost track, so put the data you expect to access most often on partition, or slice, 0.

27.3 Cylinder group

SunOS uses the Berkeley fast file system which uses **cylinder groups**. A group is formed form 32 or fewer cylinders on a disk (default 16). Each cylinder group has a redundant copy of the superblock, space for inodes, list of available blocks, and a list of data block usage within the cylinder group. Data blocks are spaced to minimize rotational delays and to keep blocks of the same file close together. By grouping cylinders in this way we reduce the amount of head movement, on average, required to access a file. The inode describing the file, and the data for the file, are likely to be in the same physical area of the disk. The position of the redundant superblock within each cylinder group is varied, so that they don't all reside on the same disk platter. This helps to insure that you can recover in the event of the loss of the primary superblock.

27.4 Physical disk structure

Disk formatted in MS-DOS has physical structure (Fig. 5).

1 sector	Boot record
The number of sectors depends on disk structure	First FAT copy
The number of sectors depends on disk structure	Second FAT copy
The number of sectors depends on disk structure	Root directory
Others rest files	Data Domain

Fig. 5. MS-DOS disk structure

1. Boot record has special program for checking is disk system or not. Checking criterion is very simple: first two files in a root directory must be IO.SYS and MSDOS.SYS. These two files must be in a root directory exactly disposed in this order. If disk is system boot record every time is disposed in first sector of zero track of zero cylinder. After physical disk sector destruction is no more in use. One physical disk division into independent logical parts for fixed disks is called section. On of the section is used for MS-DOS booting. This is an *active* section. Information about executed disk division is placed in special called *Partition table*. This is a part of *Master Boot Record*. This record places in the first sector of the zero cylinders of the zero tracks. When recorded program got control, it defines which one is active and from which cylinder this division begins.
2. First FAT copy (read about File Allocation Table)
3. Second FAT copy. FAT is very important supporting data integrity. That’s why MS-DOS has two identical tables’ copies that give opportunity “save” contents on disk after damaging using the copy for the file access.
4. Data domain. The ret space after enumeration is available for the file location. First cluster in data domain every time is cluster number two. This does not mean that all enumerated fields are in two clusters. First two FAT elements are used as disk format indicators.

28 File systems

The following are common hard disk configurations.

- **Partition** - A partition is a portion of a physical hard disk. A partition can be primary or extended
- **Primary Partition** - This is a bootable partition. One primary partition can be made active.
- **Extended Partition** - An extended partition is made from the free space on a hard disk and can be broken down into smaller logical drives. There can only be one of these per hard disk.
- **Logical Drive** - These are a primary partition or portions of an extended partition that are assigned a drive letter.
- **Volume** - This is a disk or part of a disk that is combined with space from the same or another disk to create one larger volume. This volume can be formatted and assigned a drive letter like a logical drive, but can span more than one hard disk. A volume set can be

extended without starting over, however to make it smaller, the set must be deleted and re-created.

There are various management tools that can be used to configure drives. The Disk Management MMC is a snap-in for the Computer Management Console in Windows 2000. You can create partitions, volume sets, logical drives, format disks, etc. NT 4.0 had a similar tool called the "Disk Administrator". DOS and Windows 9x utilize the FDISK utility.

When discussing Windows file systems you need to understand what File Allocation Tables (FAT) are. FAT is a table that an operating system maintains in order to map the clusters (the smallest unit of storage) that a file has been stored in. When files are written to a hard disk, the files are stored in one or more clusters that may be spread out all over the hard disk. The table allows Windows to find the "pieces" of your file and reassemble them when you wish to open it. There are several different types of file systems that are explained below:

- **FAT16** - FAT16 table entries are 16 bits in length limiting hard disk sizes to 2GB. Note that even if the OS supports larger partition sizes, the BIOS must also support logical block addressing (LBA) or the maximum partition that you will be able to create will be either 504 or 528 MB.
- **FAT32** - Created to allow more efficient use of hard drive space and allowed for partitions up to 8GB using 4KB cluster sizes. In order to format a drive as FAT32, the "Large disk Support" must be enabled when starting FDISK. FAT32 is not compatible with older versions of Windows including Windows 95A and NT. In Windows 9.x, the CVT1.EXE can be used to convert FAT16 partitions to FAT32.
- **NTFS4** - NTFS4 is the file system used by Windows NT that provides increased security and reliability over other file systems. On an NTFS partition, you can't boot from a DOS boot disk - this is one of the security features of NTFS. Additionally, a floppy disk cannot be formatted as NTFS. For this reason it might not be a bad idea to have a small partition formatted FAT so that you can boot into DOS for recovery purposes. In order to convert a FAT partition to NTFS, NT includes a utility called convert.exe.
- **NTFS5** - This is the native file system for Windows 2000. NTFS5 has many new features as follows:
  - **Encrypted File System (EFS)** - Windows 2000 NTFS volumes have the ability to encrypt data on the disk itself. Cipher.exe is a command line utility that allows for bulk or scripted file encryption.
  - **Disk Quotas** - Provides the ability to set space limitations on users on a per volume basis.
  - **Defragmentation** - Windows 2000 now includes a disk defragmenter that can be used on NTFS partitions.
  - **Volume Mount Points** - Provides the ability to add new volumes to the file system without having to assign a drive letter to them. This feature is only available on an NTFS partition using dynamic volumes.
  - **Compression** - In Windows 2000 files, folders and entire drives can be compressed by right clicking on the item to be compressed and selecting "properties" and then "advanced".

  The **convert.exe** utility can be used to convert a FAT or FAT32 partition to NTFS.
- **HPFS** - Stands for High Performance File System and is used with OS/2 operating systems. This file system can only be accessed by Windows NT 3.51 and OS/2.

Windows 9x operating systems also employ **VFAT** which is a protected-mode FAT file system that prevents DOS and the BIOS from accessing resources. VFAT is the replacement for SMARTDRV.SYS and uses a driver called **VCACHE**.

Operating System	Supported File Systems
DOS	FAT16
Windows 3.x	FAT16

Windows 95A	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows 98SE	FAT16, FAT32
Windows NT 4	FAT16, NTFS
Windows 2000	FAT16, FAT32, NTFS

In addition to the disk administration utilities previously mentioned, information about a drive can be displayed by right clicking the drive in My Computer or Windows Explorer and selecting "Properties". In a Windows 9x system, a window like the one below will appear.
Backing up drives allows you to recover your data or even the entire system if a catastrophe occurs. There are several different types of backup:

- **Full** - copies all files and marks them as being backed up.
- Incremental - copies only files created/changed since last full backup and marks them as being backed up.
- **Differential** - copies only files created/changed since last full backup and doesn’t mark them as being backed up.
- **Daily** - copies only files created/changed today and doesn’t mark them as being backed up.

In DOS backups can be run with the **BACKUP** command. There are several switches that can be added to the command.

- **/S** - Forces all files and subdirectories to be backed up.
- **/M** - Only modified files are backed up.
- **/D** - Backs up files modified after a specific date.
- **/T** - Backs up files modified after a specific time.

The Windows 98 backup utility can be accessed via **Start>Programs>Accessories>System Tools>Backup** and also via right clicking on a drive in My Computer and selecting the tools tab as previously mentioned.

There are several different hard drive utilities that can be found in the various versions of Windows that are listed below:

- **CHKDSK** - This utility is run from a DOS prompt and recovers lost allocation units on a drive that can occur when an application or the system are ended unexpectedly. The /F switch converts the lost units into a format such that the units can be viewed and deleted. Can be found in all versions of windows.
- **SCANDISK** - The ScanDisk utility inspects the hard drive for errors and corrects them. Scandisk is available in DOS 6.x and Windows 9x.
- **DEFRAG** - Reorganizes data on the disk for optimal disk performance. In DOS this utility was run from a DOS prompt. In Windows 9x and 2000 this utility can still be run from a prompt or can be accessed at Start>Programs>Accessories>System Tools>Disk Defragementer. Windows NT did not come with a defragmentation utility.
- **DRIVESPACE** - This utility for windows 9x offers many of the same features as NT's disk administrator including compression, formatting and drive information.

File Allocation Table (**FAT**) is a patented[1] file system developed by Microsoft for MS-DOS and is the primary file system for consumer versions of Microsoft Windows up to and including Windows Me.

The FAT file system is considered relatively uncomplicated, and is consequently supported by virtually all existing operating systems for personal computers. This ubiquity makes it an ideal

format for floppy disks and solid-state memory cards, and a convenient way of sharing data between disparate operating systems installed on the same computer (a dual boot environment).
The most common implementations have a serious drawback in that when files are deleted and new files written to the media, their fragments tend to become scattered over the entire media making reading and writing a slow process. Defragmentation is one solution to this, but is often a lengthy process in itself and has to be repeated regularly to keep the FAT file system clean.

	FAT12	FAT16	FAT32
Developer	Microsoft		
Full Name	File Allocation Table (12-bit version)	(16-bit version)	(32-bit version)
Introduced	1977 (Microsoft Disk BASIC)	July 1988 (MS-DOS 4.0)	August 1996 (Windows 95 OSR2)
Partition identifier	0x01 (MBR)	0x04, 0x06, 0x0E (MBR)	0x0B, 0x0C (MBR) EBD0A0A2-B9E5-4433 -87C0-68B6B72699C7 (GPT)
Structures			
Directory contents	Table		
File allocation	Linked List		
Bad blocks	Linked List		
Limits			
Max file size	32 MiB	2 GiB	4 GiB
Max number of files	4,077	65,517	268,435,437
Max filename size	8.3 or 255 characters when using LFNs		
Max volume size	32 MiB	2 GiB 4 GiB with some implementations	2 TiB
Features			
Dates recorded	Creation, modified, access		
Date range	January 1, 1980 - December 31, 2107		
Forks	Not natively		
Attributes	Read-only, hidden, system, volume label, subdirectory, archive		
Permissions	No		
Transparent compression	Per-volume, Stacker, DoubleSpace, DriveSpace		No
Transparent encryption	Per-volume only with DR-DOS		No

The FAT filesystem uses software methods that had been in use years prior to its formalization and was created by Bill Gates and Marc McDonald in 1977 for managing disks in Microsoft Disk BASIC and was incorporated by Tim Paterson in August 1980 to his 86-DOS operating system for the S-100 8086 CPU boards; the filesystem was the main difference between 86-DOS and CP/M, of which 86-DOS was otherwise mostly a clone.

29 FAT12

This initial version of FAT is now referred to as **FAT12**. As a filesystem for floppy disks, it had a number of limitations: no support for hierarchical directories, cluster addresses were "only" 12-bits long (which made the code manipulating the FAT a bit tricky) and the disk size was stored as a 16-bit count of sectors, which limited the size to 32MB.

An entry-level diskette at the time would be 5.25", single-sided, 40 tracks, with 8 sectors per track, resulting in a capacity of slightly less than 160KB. The above limits exceeded this capacity by one or more orders of magnitude and at the same time allowed all the control structures to fit inside the first track, thus avoiding head movement during read and write operations. The limits were successively lifted in the following years.

Since the sole root directory had to fit inside the first track as well, the maximum possible number of files was limited to a few dozens.

Directories

In order to properly support the newer IBM PC XT computer, which featured a 10 MB hard disk, MS-DOS 2.0 was released around the same time, at the beginning of 1983, and introduced hierarchical directories. Apart from allowing for better organisation of files, directories allowed it to store many more files on the hard disk, as the maximum number of files was no longer constrained by the (still fixed) root directory size. This number could now be equal to the number of clusters (or even greater, given that zero-sized files do not use any clusters on FAT).

The format of the FAT itself did not change. The 10 MB hard disk on the PC XT had 4 KB clusters. If a 20 MB hard disk was later installed, and formatted with MS-DOS 2.0, the resultant cluster size would be 8 KB, the boundary at 15.9 MB.

30 Initial FAT16

In 1984 IBM released the PC AT, which featured a 20 MB hard disk. Microsoft introduced MS-DOS 3.0 in parallel. Cluster addresses were increased to 16-bit, allowing for a greater number of clusters (up to 65,517) and consequently much greater filesystem sizes. However, the maximum possible number of sectors and the maximum (partition, rather than disk) size of 32 MB did not change. Therefore, although technically already "FAT16", this format was not yet what today is commonly understood under this name. A 20 MB hard disk formatted under MS-DOS 3.0, was not accessible by the older MS-DOS 2.0. Of course, MS-DOS 3.0 could still access MS-DOS 2.0 style 8 KB cluster partitions.

MS-DOS 3.0 also introduced support for high-density 1.2 MB 5.25" diskettes, which notably had 15 sectors per track, hence more space for FAT. This probably prompted a dubious optimization of the cluster size, which went down from 2 sectors to just 1. The net effect was that high density diskettes were significantly slower than older *double density* ones.

Extended partition and logical drives

Apart from improving the structure of the FAT filesystem itself, a parallel development allowing an increase in the maximum possible FAT storage space was the introduction of disk partitions. PC hard disks can only have up to 4 *primary* partitions, due to the fixed structure of the partition table in the master boot record (MBR). However, by design choice DOS would only use the partition marked as *active*, which was also the one the MBR would boot. It was not possible to create multiple primary DOS partitions using DOS tools, and third party tools would warn that such a scheme would not be compatible with DOS.

To allow the use of more partitions in a compatible way a new partition type was introduced (in MS-DOS 3.2, January 1986), the *extended partition*, which was actually just a container for additional partitions called *logical drives*. Originally only 1 logical drive was possible, allowing the use of hard-disks up to 64 MB. In MS-DOS 3.3 (August 1987) this limit was increased to 24 drives; it probably came from the compulsory letter-based C: - Z: disk naming. The logical drives were described by on-disk structures which closely resemble MBRs, probably to simplify coding, and they were chained/nested in a way analogous to Russian matryoshka dolls. Only one extended partition was allowed.

Prior to the introduction of extended partitions, some hard disk controllers (which at that time were separate option boards, since the IDE standard did not yet exist) could make large hard disks appear as two separate disks. Alternatively, special software drivers, like Ontrack's Disk Manager could be installed for the same purpose.

30.1 Final FAT16

Finally in November 1987, in Compaq DOS 3.31, came what is today called the *FAT16* format, with the expansion of the 16-bit disk sector index to 32 bits. The result was initially called the *DOS 3.31 Large File System*. Although the on-disk changes were apparently minor, the entire DOS disk code had to be converted to use 32-bit sector numbers, a task complicated by the fact that it was written in 16-bit assembly language.

In 1988 the improvement became more generally available through MS-DOS 4.0. The limit on partition size was now dictated by the 8-bit signed count of sectors-per-cluster, which had a maximum power-of-two value of 64. With the usual hard disk sector size of 512 bytes, this gives 32 KB clusters, hence fixing the "definitive" limit for the FAT16 partition size at 2 gigabytes. On magneto-optical media, which can have 1 or 2 KB sectors, the limit is proportionally greater.

Much later, Windows NT increased the maximum cluster size to 64 KB by considering the sectors-per-cluster count as unsigned. However, the resulting format was not compatible with any other FAT implementation of the time, and anyway, generated massive internal fragmentation. Windows 98 also supported reading and writing this variant, but its disk utilities didn't work with it.

30.2 Long File Names (VFAT, LFNs)

One of the "user experience" goals for the designers of Windows 95 was the ability to use long file names (LFNs), in addition to classic 8.3 names. LFNs were implemented using a work-around in the way directory entries are laid out (see below). The version of the file system with this extension is usually known as VFAT after the Windows 95 VxD device driver.

Interestingly, the VFAT driver actually appeared before Windows 95, in Windows for Workgroups 3.11, but was only used for implementing 32-bit File Access, a higher performance protected mode file access method, bypassing DOS and directly using either the BIOS, or, better, the Windows-native protected mode disk drivers. It was a backport; Microsoft's ads for WfW 3.11 said 32-bit File Access was based on "the 32-bit file system from our Chicago project".

In Windows NT, support for long file names on FAT started from version 3.5.

30.3 FAT32

In order to overcome the volume size limit of FAT16, while still allowing DOS real-mode code to handle the format without unnecessarily reducing the available conventional memory, Microsoft decided to implement a newer generation of FAT, known as **FAT32**, with cluster counts held in a 32-bit field, of which 28 bits are currently used.

In theory, this should support a total of approximately 268,435,438 ($< 2^{28}$) clusters, allowing for drive sizes in the range of 2 terabytes. However, due to limitations in Microsoft's scandisk utility, the FAT is not allowed to grow beyond 4,177,920 ($< 2^{22}$) clusters, placing the volume limit at 124.55 gigabytes, unless "scandisk" is not needed.

FAT32 was introduced with Windows 95 OSR2, although reformatting was needed to use it, and DriveSpace 3 (the version that came with Windows 95 OSR2 and Windows 98) never

supported it. Windows 98 introduced a utility to convert existing hard disks from FAT16 to FAT32 without loss of data. In the NT line, support for FAT32 arrived in Windows 2000.

Windows 2000 and Windows XP can read and write to FAT32 filesystems of any size, but the format program on these platforms can only create FAT32 filesystems up to 32 GB. Thompson and Thompson (2003) write[4] that "Bizarrely, Microsoft states that this behavior is by design." Microsoft's knowledge base article 184006[3] indeed confirms the limitation and the by design statement, but gives no rationale or explanation. Peter Norton's opinion[5] is that "Microsoft has intentionally crippled the FAT32 file system."

The maximum possible size for a file on a FAT32 volume is 4 GiB minus 1 B (2^{32}-1 bytes). For most users, this has become the most nagging limit of FAT32 as of 2005, since video capture and editing applications can easily exceed this limit, as can the system swap file.

Third party support

The alternative IBM PC operating systems — such as Linux, FreeBSD, and BeOS — have all supported FAT, and most added support for VFAT and FAT32 shortly after the corresponding Windows versions were released. Early Linux distributions also supported a format known as UMSDOS, which was FAT with Unix file attributes (such as long file name and access permissions) stored in a separate file called --linux-.---. UMSDOS fell into disuse after VFAT was released and is not enabled by default in Linux kernels from version 2.5.7 onwards [5]. The Mac OS X operating system also supports the FAT filesystems on volumes other than the boot disk.

30.4 FAT and Alternate Data Streams

The FAT filesystem itself is not designed for supporting ADS, but some operating systems that heavily depend on them have devised various methods for handling them in FAT drives. Such methods either store the additional information in extra files and directories (Mac OS), or give new semantics to previously unused fields of the FAT on-disk data structures (OS/2 and Windows NT). The second design, while presumably more efficient, prevents any copying or backing-up of those volumes using non-aware tools; manipulating such volumes using non-aware disk utilities (e.g. defragmenters or CHKDSK) will probably lose the information.

Mac OS using PC Exchange stores its various dates, file attributes and long filenames in a hidden file called FINDER.DAT, and Resource Forks (a common Mac OS ADS) in a subdirectory called RESOURCE.FRK, in every directory where they are used. From PC Exchange 2.1 onwards, they store the Mac OS long filenames as standard FAT long filenames and convert FAT filenames longer than 31 characters to unique 31-character filenames, which can then be made visible to Macintosh applications.

Mac OS X stores metadata (Resource Forks, file attributes, other ADS) in a hidden file with a name constructed from the owner filename prefixed with "._", and Finder stores some folder and file metadata in a hidden file called ".DS_Store".

OS/2 heavily depends on extended attributes (EAs) and stores them in a hidden file called "EA DATA. SF" in the root directory of the FAT12 or FAT16 volume. This file is indexed by 2 previously reserved bytes in the file's (or directory's) directory entry. In the FAT32 format, these bytes hold the upper 16 bits of the starting cluster number of the file or directory, hence making it difficult to store EAs on FAT32. Extended attributes are accessible via the Workplace Shell desktop, through REXX scripts, and many system GUI and command-line utilities (such as 4OS2).
Windows NT supports the handling of extended attributes in HPFS, NTFS, and FAT. It stores EAs on FAT using exactly the same scheme as OS/2, but does not support any other kind of ADS as held on NTFS volumes. Trying to copy a file with any ADS other than EAs from an NTFS volume to a FAT volume gives a warning message with the names of the ADSs that will be lost.

Windows 2000 onward acts exactly as Windows NT, except that it ignores EAs when copying to FAT32 without any warning (but shows the warning for other ADSs, like "Macintosh Finder Info" and "Macintosh Resource Fork").

Future

Microsoft has recently secured patents for VFAT and FAT32 (but not the original FAT), which is causing concern that the company might later seek royalties from Linux distros and from media vendors that pre-format their products (see FAT Licensing below). Despite two earlier rulings against them, Microsoft prevailed and was awarded the patents.

Since Microsoft has announced the discontinuation of its MS-DOS-based consumer operating systems with Windows Me, it remains unlikely that any new versions of FAT will appear. For most purposes, the NTFS file system that was developed for the Windows NT line is superior to FAT from the points of view of efficiency, performance and reliability; its main drawbacks are the size overhead for small volumes and the very limited support by anything other than the NT-based versions of Windows, since the exact specification is a trade secret of Microsoft, which in turn makes it difficult to use a DOS floppy for recovery purposes. Microsoft provided a recovery console to work around this issue, but for security reasons it severely limited what could be done through the Recovery Console by default.

FAT is still the normal filesystem for removable media (with the exception of CDs and DVDs), with FAT12 used on floppies, and FAT16 on most other removable media (such as flash memory cards for digital cameras and USB flash drives). Most removable media is not yet large enough to benefit from FAT32, although some larger flash drives do make use of it. FAT is used on these drives for reasons of compatibility and size overhead, as well as the fact that file permissions on removable media are likely to be more trouble than they are worth.

The FAT32 formatting support in Windows 2000 and XP is limited to drives of 32 gigabytes, which effectively forces users of modern hard drives either to use NTFS or to format the drive using third party tools such as a port of mkdosfs or fat32format.

30.5 Main disk structures

Master Boot Record	File Allocation Table #1	File Allocation Table #2	Root Directory	All Other Data ... The Rest of the Disk

A FAT file system is composed of four different sections.

1. The **Reserved sectors**, located at the very beginning. The first reserved sector is the Boot Sector (aka *Partition Boot Record*). It includes an area called the *BIOS Parameter Block* (with some basic file system information, in particular its type, and pointers to the location of the other sections) and usually contains the operating system's boot loader code. The total count of reserved sectors is indicated by a field inside the Boot Sector. Important information from the Boot Sector is accessible through an operating system structure called the *Drive Parameter Block* in DOS and OS/2.
2. The **FAT Region**. This contains two copies of the *File Allocation Table* for the sake of redundancy, although the extra copy is rarely used, even by disk repair utilities. These are maps of the partition, indicating how the clusters are allocated.
3. The **Root Directory Region**. This is a *Directory Table* that stores information about the files and directories in the root directory. With FAT32 it can be stored anywhere in the partition, however with earlier versions it is always located immediately after the *FAT Region*.
4. The **Data Region**. This is where the actual file and directory data is stored and takes up most of the partition. The size of files and subdirectories can be increased arbitrarily (as long as there are free clusters) by simply adding more links to the file's chain in the FAT. Note however, that each cluster can be taken only by one file, and so if a 1 KB file resides in a 32 KB cluster, 31 KB are wasted.

Boot Sector

The Boot Sector is of the following format:

<table>
<tr><th>Byte Offset</th><th>Length (bytes)</th><th>Description</th></tr>
<tr><td>0x00</td><td>3</td><td>Jump instruction (to skip over header on boot)</td></tr>
<tr><td>0x03</td><td>8</td><td>OEM Name (padded with spaces). MS-DOS checks this field to determine which other parts of the boot record can be relied on [6] [7]. Common values are IBM 3.3 (with two spaces between the "IBM" and the "3.3") and MSDOS5.0.</td></tr>
<tr><td>0x0b</td><td>2</td><td>Bytes per sector. The BIOS Parameter Block starts here.</td></tr>
<tr><td>0x0d</td><td>1</td><td>Sectors per cluster</td></tr>
<tr><td>0x0e</td><td>2</td><td>Reserved sector count (including boot sector)</td></tr>
<tr><td>0x10</td><td>1</td><td>Number of file allocation tables</td></tr>
<tr><td>0x11</td><td>2</td><td>Maximum number of root directory entries</td></tr>
<tr><td>0x13</td><td>2</td><td>Total sectors (if zero, use 4 byte value at offset 0x20)</td></tr>
<tr><td>0x15</td><td>1</td><td>Media descriptor
<table>
<tr><td>0xF8</td><td>Single sided, 80 tracks per side, 9 sectors per track</td></tr>
<tr><td>0xF9</td><td>Double sided, 80 tracks per side, 9 sectors per track</td></tr>
<tr><td>0xFA</td><td>Single sided, 80 tracks per side, 8 sectors per track</td></tr>
<tr><td>0xFB</td><td>Double sided, 80 tracks per side, 8 sectors per track</td></tr>
<tr><td>0xFC</td><td>Single sided, 40 tracks per side, 9 sectors per track</td></tr>
<tr><td>0xFD</td><td>Double sided, 40 tracks per side, 9 sectors per track</td></tr>
<tr><td>0xFE</td><td>Single sided, 40 tracks per side, 8 sectors per track</td></tr>
<tr><td>0xFF</td><td>Double sided, 40 tracks per side, 8 sectors per track</td></tr>
</table>
Same value of media descriptor should be repeated as first byte of each copy of FAT. Certain operating systems (MSX-DOS version 1.0) ignore boot sector parameters altogether and use media descriptor value from the first byte of FAT to determine filesystem parameters.</td></tr>
<tr><td>0x16</td><td>2</td><td>Sectors per file allocation table (FAT16)</td></tr>
<tr><td>0x18</td><td>2</td><td>Sectors per track</td></tr>
<tr><td>0x1a</td><td>2</td><td>Number of heads</td></tr>
<tr><td>0x1c</td><td>4</td><td>Hidden sectors</td></tr>
<tr><td>0x20</td><td>4</td><td>Total sectors (if greater than 65535; see offset 0x13)</td></tr>
<tr><td>0x24</td><td>4</td><td>Sectors per file allocation table (FAT32). The Extended BIOS Parameter Block starts here.</td></tr>
<tr><td>0x24</td><td>1</td><td>Physical drive number (FAT16)</td></tr>
<tr><td>0x25</td><td>1</td><td>Current head (FAT16)</td></tr>
<tr><td>0x26</td><td>1</td><td>Signature (FAT16)</td></tr>
<tr><td>0x27</td><td>4</td><td>ID (FAT16)</td></tr>
<tr><td>0x2b</td><td>11</td><td>Volume Label</td></tr>
<tr><td>0x36</td><td>8</td><td>FAT file system type (e.g. FAT, FAT12, FAT16, FAT32)</td></tr>
</table>

0x3e	448	Operating system boot code
0x1FE	2	End of sector marker (0x55 0xAA)

The boot sector is portrayed here as found on e.g. an OS/2 1.3 boot diskette. Earlier versions used a shorter BIOS Parameter Block and their boot code would start earlier (for example at offset 0x2b in OS/2 1.1).

30.5.1.1 Exceptions

The implementation of FAT used in MS-DOS for the Apricot PC had a different boot sector layout, to accommodate that computer's non-IBM compatible BIOS. The jump instruction and OEM name were omitted, and the MS-DOS filesystem parameters (offsets 0x0B - 0x17 in the standard sector) were located at offset 0x50. Later versions of Apricot MS-DOS gained the ability to read and write disks with the standard boot sector in addition to those with the Apricot one.

DOS Plus on the BBC Master 512 did not use conventional boot sectors at all. Data disks omitted the boot sector and began with a single copy of the FAT (the first byte of the FAT was used to determine disk capacity) while boot disks began with a miniature ADFS filesystem containing the boot loader, followed by a single FAT. It could also access standard PC disks formatted to 180 KB or 360 KB, again using the first byte of the FAT to determine capacity.

31 File Allocation Table

A partition is divided up into identically sized **clusters**, small blocks of contiguous space. Cluster sizes vary depending on the type of FAT file system being used and the size of the partition, typically cluster sizes lie somewhere between 2 KB and 32 KB. Each file may occupy one or more of these clusters depending on its size; thus, a file is represented by a chain of these clusters (referred to as a singly linked list). However these chains are not necessarily stored adjacent to one another on the disk's surface but are often instead *fragmented* throughout the Data Region.

The **File Allocation Table** (**FAT**) is a list of entries that map to each cluster on the partition. Each entry records one of five things:

- the address of the next cluster in a chain
- a special *end of file* (*EOF*) character that indicates the end of a chain
- a special character to mark a bad cluster
- a special character to mark a reserved cluster
- a zero to note that that cluster is unused

Each version of the FAT file system uses a different size for FAT entries. The size is indicated by the name, for example the FAT16 file system uses 16 bits for each entry while the FAT32 file system uses 32 bits. This difference means that the File Allocation Table of a FAT32 system can map a greater number of clusters than FAT16, allowing for larger partition sizes with FAT32. This also allows for more efficient use of space than FAT16, because on the same hard drive a FAT32 table can address smaller clusters which means less wasted space.

FAT entry values:

FAT12	FAT16	FAT32	Description
0x000	0x0000	0x?0000000	Free Cluster
0x001	0x0001	0x?0000001	Reserved Cluster
0x002 - 0xFEF	0x0002 - 0xFFEF	0x?0000002 - 0x?FFFFFEF	Used cluster; value points to next cluster
0xFF0 - 0xFF6	0xFFF0 - 0xFFF6	0x?FFFFFF0 -	Reserved values

		0x?FFFFFF6	
0xFF7	0xFFF7	0x?FFFFFF7	Bad cluster
0xFF8 - 0xFFF	0xFFF8 - 0xFFFF	0x?FFFFFF8 - 0x?FFFFFFF	Last cluster in file

Note that FAT32 uses only 28 bits of the 32 possible bits. The upper 4 bits are usually zero but are reserved and should be left untouched. In the table above these are denoted by a question mark.

The first cluster of the data area is cluster #2. That leaves the first two entries of the FAT unused. In the first byte of the first entry a copy of the media descriptor is stored. The remaining bits of this entry are 1. In the second entry the end-of-file marker is stored. The high order two bits of the second entry are sometimes, in the case of FAT16 and FAT32, used for dirty volume management: high order bit 1: last shutdown was clean; next highest bit 1: during the previous mount no disk I/O errors were detected.[6]

32 Floppy disk

Fig. 6. Floppy disk

A **floppy disk** is a data storage device that is composed of a ring of thin, flexible (i.e. "floppy") magnetic storage medium encased in a square or rectangular plastic wallet. Floppy disks are read and written by a **floppy disk drive** or **FDD**, the latter initialism not to be confused with "fixed disk drive", which is an old IBM term for a hard disk drive.

Floppy disk format	**Year introduced**	**Storage capacity (binary kilobytes if not stated)**	**Marketed capacity[1]**
8-inch (read-only)	1969	80	←
8-inch	1972	183.1	1.5 Megabit
8-inch	1973	256	256 KB
8-inch DD	1976	500	0.5 MB
5¼-inch (35 track)	1976	89.6	110 KB
8-inch double sided	1977	1200	1.2 MB
5¼-inch DD	1978	360	360 KB
3½-inch HP single sided	1982	280	264 KB
3-inch	1982[j]	360	←
3½-inch (DD at release)	1984	720	720 KB
5¼-inch QD	1984	1200	1.2 MB
3-inch DD	1984	720	←
3-inch Mitsumi Quick Disk	1985	128 to 256	←

<table>
<tr><td>2-inch</td><td>1985</td><td>720</td><td>←</td></tr>
<tr><td>5¼-inch Perpendicular</td><td>1986]</td><td>100 MiB</td><td>←</td></tr>
<tr><td>3½-inch HD</td><td>1987</td><td>1440</td><td>1.44 MB</td></tr>
<tr><td>3½-inch ED</td><td>1991</td><td>2880</td><td>2.88 MB</td></tr>
<tr><td>3½-inch LS-120</td><td>1996</td><td>120.375 MiB</td><td>120 MB</td></tr>
<tr><td>3½-inch LS-240</td><td>1997</td><td>240.75 MiB</td><td>240 MB</td></tr>
<tr><td>3½-inch HiFD</td><td>1998/99</td><td>150/200 MiB]</td><td>150/200 MB</td></tr>
<tr><td colspan="4">Acronyms: DD = Double Density; QD = Quad Density; HD = High Density ED = Extended Density; LS = Laser Servo; HiFD = High capacity Floppy Disk</td></tr>
<tr><td colspan="4">[1]The marketed capacities of floppy disks frequently corresponded only vaguely to their their actual storage capacities; the 1.44 MB value for the 3½-inch HD floppies is the most widely known example. See reported storage capacity.</td></tr>
<tr><td colspan="4">Dates and capacities marked ? are of unclear origin and need source information; other listed capacities refer to:
For 8-inch: standard IBM formats as used by the System/370 mainframes and newer systems
For 5¼- and 3½-inch: standard PC formats, capacities quoted are the total size of all sectors on the disk and include space used for the bootsector and filesystem
Other formats may get more or less capacity from the same drives and disks.</td></tr>
</table>

33 AUTOEXEC.BAT

File found on the MS-DOS operating system. It is a plain-text batch file that is located in the root directory of the boot device.

Usage

AUTOEXEC.BAT is only used on MS-DOS or Microsoft Windows versions based on MS-DOS, such as Windows 3.x, Windows 95, Windows 98, and Windows Me. The file is executed once the operating system has booted and after the CONFIG.SYS file has been processed. On Windows, this occurs before the graphical environment has been started.

AUTOEXEC.BAT is most often used to set environment variables and run virus scanners, system enhancements, utilities, and driver handlers that must operate at the lowest level possible. Applications that run within the Windows environment upon its loading are listed in the Windows registry.

Lines prefixed with the string "REM" are remarks and are not run as part of AUTOEXEC.BAT. The "REM" lines are used for comments or to disable drivers (say, for a CD-ROM).

34 CONFIG.SYS

CONFIG.SYS is the primary configuration file for the MS-DOS and OS/2 operating systems. It is a special file that contains setup or configuration instructions for the computer system. The commands in this file configure DOS for use with devices and applications in the system. The commands also set up the memory managers in the system. After processing the CONFIG.SYS file, DOS proceeds to load and execute the command shell specified in the shell= line of CONFIG.SYS, or COMMAND.COM if there is no such line. The command shell in turn is responsible for processing the AUTOEXEC.BAT file.

The system can still boot if these files are missing or corrupted. However, these two files are essential for the complete bootup process to occur with the DOS operating system. They contain

information that is used to change the operating system for personal use. They also contain the requirements of different software application packages. A DOS system would require troubleshooting if either of these files became damaged or corrupted.

CONFIG.SYS is composed mostly of name=value statements which look like variable assignments. In fact these will either define some tunable parameters often resulting in reservation of memory, or load files, mostly TSRs and device drivers, into memory.

In DOS, CONFIG.SYS is located in the root directory of the drive from which DOS was booted. In some versions of DOS it may have an alternate filename, e.g. **FDCONFIG.SYS** in FreeDOS, or **DCONFIG.SYS** in some versions of DR-DOS.

Both CONFIG.SYS and AUTOEXEC.BAT can still be found included in the system files of the later Microsoft Windows operating systems. Usually these files are empty files, with no content. OS/2 did not use the autoexec.bat file, instead using startup.cmd.

In the OS/2 subsystem of Windows NT, what appeared as CONFIG.SYS to OS/2 programs was actually stored in the registry.

34.1 Example CONFIG.SYS file for DOS

device = c:\dos\himem.sys
device = c:\dos\emm386.exe umb
dos = high,umb
devicehigh = c:\windows\mouse.sys
devicehigh = c:\dos\setver.exe
devicehigh = c:\dos\smartdrv.exe
country = 044,437,c:\dos\country.sys
shell = c:\dos\command.com c:\dos /e:512 /p

Windows NT

On Windows NT and its derivatives, Windows 2000 and Windows XP, the equivalent file is called AUTOEXEC.NT and is located in the %SystemRoot%\system32 directory. The file is not used during the operating system boot process; it is executed when the MS-DOS environment is started, which occurs when an MS-DOS application is loaded.

The AUTOEXEC.BAT file may often be found on Windows NT, in the root directory of the boot drive. Windows only considers the "SET" statements which it contains, in order to define environment variables global to all users. Setting environment variables through this file may be interesting if for example MS-DOS is also booted from this drive (this requires that the drive be FAT) or to keep the variables across a reinstall. This is an exotic usage today so this file remains almost always empty. The TweakUI applet from the *PowerToys* collection allows to control this feature (*Parse Autoexec.bat at logon*).

35 Computer software

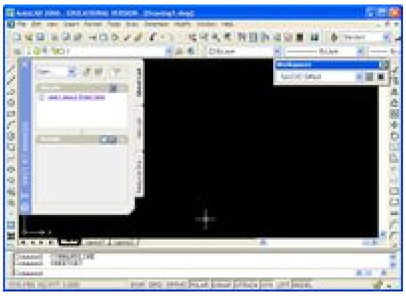


Fig. 7. A screenshot of computer software in action.

The various programs by which a computer controls aspects of its operations, such as those for translating data from one form to another.

Computer software (or simply **software**) refers to any of the various programs by which a computer controls aspects of its operations, such as those for translating data from one form to another, as contrasted with hardware, which is the physical equipment comprising the installation.

The term is roughly synonymous with computer program but is more generic in scope. In most computers, the moment-to-moment control of the machine resides in a special software program called an operating system, or supervisor. Other forms of software include assemblers and compilers for programming languages and applications for business and home use (see computer program). Software is of great importance; the usefulness of a highly sophisticated array of hardware can be severely compromised by the lack of adequate software.

The term "software" was first used in this sense by John W. Tukey in 1957. In computer science and software engineering, **computer software** is all information processed by computer systems, programs and data. The concept of reading different sequences of instructions into the memory of a device to control computations was invented by Charles Babbage as part of his difference engine. The theory that is the basis for most modern software was first proposed by Alan Turing in an essay.

35.1 Relationship to hardware

Computer software is so called in contrast to computer hardware, which encompasses the physical interconnections and devices required to store and execute (or run) the software. In computers, software is loaded into RAM and executed in the central processing unit. At the lowest level, software consists of a machine language specific to an individual processor. A machine language consists of groups of binary values (which may be represented by octal or hexadecimal numerals) signifying processor instructions (object code), which change the state of the computer from its preceding state. Software is an ordered sequence of instructions for changing the state of the computer hardware in a particular sequence. It is generally written in high-level programming languages that are easier and more efficient for humans to use (closer to natural language) than machine language. High-level languages are compiled or interpreted into machine language object code. Software may also be written in an assembly language, essentially, a mnemonic representation of a machine language using a natural language alphabet. Assembly language must be assembled into object code via an assembler.

35.2 Relationship to data

Software has historically been considered an intermediary between electronic hardware and *data,* which later the *hardware* processes according to the sequence of instructions defined by the software. As computational math becomes increasingly complex, the distinction between software and data becomes less precise. Data has generally been considered as either the output or input of executed software. However, data is not the only possible output or input. For example, (system) configuration information may also be considered input, although not *necessarily* considered data (and certainly not applications data). The output of a particular piece of executed software may be the input for another executed piece of software. Therefore, software may be considered an interface between hardware, data, and/or (other) software.

36 System, programming and application software

Practical computer systems divide software into three major classes: system software, programming software and application software, although the distinction is somewhat arbitrary, and often blurred.

System software helps run the computer hardware and computer system. It includes operating systems, device drivers, diagnostic tools, servers, windowing systems, utilities and more. The purpose of systems software is to insulate the applications programmer as much as possible from the details of the particular computer complex being use, especially memory and other hardware features, and such accessory devices as communications, printers, readers, displays, keyboards, etc.

Programming software usually provides tools to assist a programmer in writing computer programs and software using different programming languages in a more convenient way. The tools include text editors, compilers, interpreters, linkers, debuggers, and so on. An Integrated

development environment (IDE) merges those tools into a software bundle, and a programmer may not need to type multiple commands for compiling, interpreter, debugging, tracing, and etc., because the IDE usually has an advanced *graphical user interface,* or GUI.

Application software allows humans to accomplish one or more specific (non-computer related) tasks. Typical applications include industrial automation, business software, educational software, medical software, databases and computer games. Businesses are probably the biggest users of application software, but almost every field of human activity (from **a**-bombs to **z**ymurgy) now uses some form of application software. It is used to automate all sorts of functions. Many examples may be found at the Business Software Directory.

36.1 Software program and library

A software program may not be sufficiently complete for execution by a computer. In particular, it may require additional software from a software library in order to be complete. Such a library may include software components used by stand-alone programs, but which cannot be executed on their own. Thus, programs may include standard routines that are common to many programs, extracted from these libraries. Libraries may also *include* 'stand-alone' programs which are activated by some computer event and/or perform some function (e.g., of computer 'housekeeping') but do not return data to their activating program. Programs may be called by other programs and/or may call other programs.

37 Three layers of software

Starting in the 1980s, application software has been sold in mass-produced packages through retailers

Users often see things differently than programmers. People who use modern general purpose computers (as opposed to embedded systems, analog computers, supercomputers, etc.) usually see three layers of software performing a variety of tasks: platform, application, and user software.

Platform software

Platform includes the basic input-output system (often described as *firmware* rather than *software*), device drivers, an operating system, and typically a graphical user interface which, in total, allow a user to interact with the computer and its peripherals (associated equipment). Platform software often comes bundled with the computer, and users may not realize that it exists or that they have a choice to use different platform software.

Application software

Application software or Applications are what most people think of when they think of software. Typical examples include office suites and video games. Application software is often purchased separately from computer hardware. Sometimes applications are bundled with the computer, but that does not change the fact that they run as independent applications. Applications are almost always independent programs from the operating system, though they are often tailored for specific platforms. Most users think of compilers, databases, and other "system software" as applications.

User-written software

User software tailors systems to meet the users specific needs. User software include spreadsheet templates, word processor macros, scientific simulations, graphics and animation scripts. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is. Depending on how competently the user-written software has been integrated into purchased application packages, many users may not be aware of the distinction between the purchased packages, and what has been added by fellow co-workers.

38 Software operation

Computer software has to be "loaded" into the computer's storage (also known as *memory* and *RAM*).

Once the software is loaded, the computer is able to operate the software. Computers operate by *executing* the computer program. This involves passing instructions from the application software, through the system software, to the hardware which ultimately receives the instruction as machine code. Each instruction causes the computer to carry out an operation -- moving data, carrying out a computation, or altering the control flow of instructions.

Data movement is typically from one place in memory to another. Sometimes it involves moving data between memory and registers which enable high-speed data access in the CPU. Moving data, especially large amounts of it, can be costly. So, this is sometimes avoided by using "pointers" to data instead. Computations include simple operations such as incrementing the value of a variable data element. More complex computations may involve many operations and data elements together.

Instructions may be performed sequentially, conditionally, or iteratively. Sequential instructions are those operations that are performed one after another. Conditional instructions are performed such that different sets of instructions execute depending on the value(s) of some data. In some languages this is know as an "if" statement. Iterative instructions are perfomed repetitively and may depend on some data value. This is sometimes called a "loop." Often, one instruction may "call" another set of instructions that are defined in some other program or module. When more than one computer processor is used, instructions may be executed simultaneously.

A simple example of the way software operates is what happens when a user selects an entry such as "Copy" from a menu. In this case, a conditional instruction is executed to copy text from data in a document to a clipboard data area. If a different menu entry such as "Paste" is chosen, the software executes the instructions to copy the text in the clipboard data area to a place in the document.

Depending on the application, even the example above could become complicated. The field of software engineering endeavors to manage the complexity of how software operates. This is especially true for software that operates in the context of a large or powerful computer system.
Kinds of software by operation: computer program as executable, source code or script, configuration.

39 Memory control drivers

SMARTDRV.SYS is a disk caching program for DOS and Windows 3.x systems. The smartdrive program keeps a copy of recently accessed hard disk data in memory. When a program or MSDOS reads data, smartdrive first checks to see if it already has a copy and if so supplies it instead of reading from the hard disk.

Memory Management

- First 640k is **Conventional Memory**
- 640k to 1024k is **Upper Memory**
- Above 1024k is **Extended Memory**
- **HIMEM.SYS** is loaded in CONFIG.SYS as the first driver to manage the Extended Memory are and to convert this to XMS (Extended Memory Specification). The first 64k of extended memory has been labeled High Memory (HMA). DOS can be put here by putting DOS=HIGH in CONFIG.SYS.
- **EMM386.EXE** is loaded in CONFIG.SYS after HIMEM.SYS has been successfully loaded. This is used in the hardware reserved 384k of space in upper memory (640k-1024k) and creates EMS (Extended Memory Specification).
- **Virtual Memory** relies upon EMS (therefore EMM386.EXE) and uses hard disk space as memory.
- **Memory control commands**

40 Cash memory

DISK CASHING

To help understand the theory of caching, visualize an old, hand-operated water pump. Each stroke of the pump's handle delivers a set amount of water into a glass. It may take two or three handle strokes to fill a glass. Now, visualize several glasses that need to be filled. You are constantly pumping the handle to keep up with the demand. Next, introduce a holding tank. With this, instead of the water going directly into a glass, it goes into the tank. The advantage is, once the holding tank is filled, constant pumping is not required to keep up with the demand.

Disk caching may be thought of as an electronic version of a holding tank. With *MS-DOS* version 5.0, the holding tank is built in with Smartdrv.sys.

Cache: A bank of high-speed memory set aside for frequently accessed data. The term "*cashing*" describes placing data in the cache. Memory caching and disk cashing are the two most common methods used by PCs.

Keeping the most frequently used disk sectors in operational memory (hereafter – RAM) is called **disk cashing**. It is used to increase speed of information exchange between the hard disk and RAM. It’s well known that the relatively low speed of information exchange between these two devices used to be one of the weakest points limiting computer productivity. No doubt, there are some other weak points, for example, information exchange between fast microprocessor and slow RAM, but, as long as DOS doesn’t suggest any ways of dealing with such problems, we are not going to consider them.

To perform disk cashing, a special buffer region called **cash** is organized in RAM. It works as a canal for information exchange and is operated by the resident program called **cash administrator**.

Read data is placed into cash and kept there until another new portion of data replaces it. When the specific data is required it can be retrieved from the fast cash and there will be no need of reading it from the disk. So, the speed of data reading from the “disk” increases. This procedure is called “end-to-end reading”.

Even more noticeable effect is achieved by preliminary reading (without operational system’s request) of data and placing it into cash, because this operation can be fulfilled without microprocessor being idle, that is asynchronically.

Most of the modern cash administrators provide cashing not only for reading, but also for writing. **Write cashing** is used when placing data on disk after operational system’s instructions. Firstly, this data is placed into cash and then, when it is “convenient” for a PC, placed to disk, so that the real writing into disk is organized asynchronically. Further we will call this process an “intermediate writing”. After writing data into cash instead of writing it to the disk, DOS is notified about the end of writing operation. Since it is accomplished much faster than writing straight to the disk, write cashing is very effective. This effect is even more noticeable when executing such operations:

renewal of the data recently written to the disk (in the case of cashing it may be refreshed in RAM)

using (repeatedly reading) recently written to disk data (because of cashing the writing process can be done without reading data from the disk).

Besides increasing the productivity of the PC, the disk cashing allows to increase the working lifetime of the hard disk due to the reduction of the disk wear.

Disk cashing successfully combines the positive points of the I/O buffering and of the virtual disk usage, because (analogically to virtual disk) it provides storing of big amounts of data in RAM and (analogically to I/O buffers) keeps only the frequently used data. This allows minimizing the need of allocating large amounts of RAM to be used as a buffer. Besides, cashing (analogically to I/O buffering) is totally “transparent” for the users and programs, when using a virtual disk user must copy files to it by himself, and after that, probably, has to configure the programs, that will use those files. However, cash administrator is usually much bigger than the virtual disk driver due to the amount of operations it has to perform. This can cause user to stop using disk cashing.

DOS simply can’t do without I/O buffering, which represents the *simplified* variant of cashing. That’s why it is represented with the compact code: anticipatory reading and intermediate writing aren’t executed. However the purpose of I/O buffering is not just to minimize the access time to the same data, but also to extract the logical records from the physical records and vice-versa – to form physical records from the logical records. **Physical record** is a portion of data, which is transferred between RAM and external memory (for disks – contents of a sector). **Logical record** is a portion of data, inquired by a program or outputted by it. The tools of the I/O buffering allow reading of the physical record for only one time, even if there are several logical records in it needed for a certain program’s performance. Analogically, physical record is written to disk only after it is formed from several logical records. Without the I/O buffering tools the reading of each logical record (even from the same sector) would cause the frequent reading of this sector from the disk. As for the output of each logical record, it would require the operation of writing the whole physical record to disk, moreover after it’s anticipatory reading and refreshing. All these operations, in addition to the significant waste of time, would require additional efforts of the programmers.

Since I/O buffering tools perform blocking and unblocking of physical records, cashing tools are used for organizing work with physical records (in case of disks – the contents of the sectors).

Disk Caching With MS-DOS' Smartdrv.sys

Total system performance is a composite of several factors. Two main factors are central processing unit (CPU) type and speed, and hard drive access time. Other factors in the mix are the software programs themselves. Certain programs, like databases and some computer-aided design (CAD) packages, constantly access your hard drive by opening and closing files. Since the hard drive is a mechanical device with parts like read/write heads that physically access data, this constant access slows things down. Short of buying faster equipment, changing the way data is transferred to the CPU is the most effective way to speed up your system. This can be done with **disk caching** (pronounced disk "cashing").

41 UNIX operation system. Main features and commands. UNIX / Linux

UNIX is one of the very oldest operating systems in the computer world, and is still widely used today. However, it is not a very *conspicuous* operating system. Somewhat arcane in its operation and interface, it is ideally suited for the needs of large enterprise computing systems. It is also the most common operating system run by servers and other computers that form the bulk of the Internet. While you may never use UNIX on your local PC, you are using it indirectly, in one form or another, every time you log on to the 'net.

While few people run UNIX on their own systems, there are in fact a number of different versions of UNIX available for the PC, and millions of PC users have chosen to install "UNIXy" operating systems on their own desktop machines. There are dozens of variants of the basic UNIX interface; the most popular one for the PC platform is *Linux*, which is itself available in many flavors. While UNIX operating systems can be difficult to set up and require some knowledge to operate, they are very stable and robust, are efficient with system resources--and are generally free or very inexpensive to obtain.

UNIX operating systems are designed to use the "UNIX file system". I put that phrase in quotes, because there is no single UNIX file system, any more than there is a single UNIX *operating* system. However, the file systems used by most of the UNIX operating system types out there are fairly similar, and rather distinct from the file systems used by other operating systems, such as DOS or Windows.

As an operating system geared specifically for use on the PC, Linux is the UNIX variant that gets the most attention in PC circles. To improve its appeal, the programmers who are continually working to update and improve Linux have put into the operating system compatibility support for most of the other operating systems out there. Linux will read and write to FAT partitions, and with newer versions this includes FAT32.

Unix (officially trademarked as **UNIX®**) is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs including Ken Thompson, Dennis Ritchie and Douglas McIlroy. Today's Unix systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organizations.

As of 2007, the owner of the trademark *UNIX®* is The Open Group, an industry standards consortium. Only systems fully compliant with and certified to the Single UNIX Specification qualify as "UNIX®" (others are called "Unix system-like" or "Unix-like").

During the late 1970s and early 1980s, Unix's influence in academic circles led to large-scale adoption of Unix (particularly of the BSD variant, originating from the University of California, Berkeley) by commercial startups, the most notable of which is Sun Microsystems. Today, in addition to certified Unix systems, Unix-like operating systems such as Linux and BSD derivatives are commonly encountered.

Sometimes, "traditional Unix" may be used to describe a Unix or an operating system that has the characteristics of either Version 7 Unix or UNIX System V.

41.1 Overview

Unix operating systems are widely used in both servers and workstations. The Unix environment and the client-server program model were essential elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers.

Both Unix and the C programming language were developed by AT&T and distributed to government and academic institutions, causing both to be ported to a wider variety of machine families than any other operating system. As a result, Unix became synonymous with "open systems".

Unix was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication

(IPC) as files; and the use of a large number of small programs that can be strung together through a command line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are known as the Unix philosophy.

Under Unix, the "operating system" consists of many of these utilities along with the master control program, the kernel. The kernel provides services to start and stop programs, handle the file system and other common "low level" tasks that most programs share, and, perhaps most importantly, schedules access to hardware to avoid conflicts if two programs try to access the same resource or device simultaneously. To mediate such access, the kernel was given special rights on the system and led to the division between *user-space* and *kernel-space*.

The microkernel tried to reverse the growing size of kernels and return to a system in which most tasks were completed by smaller utilities. In an era when a "normal" computer consisted of a hard disk for storage and a data terminal for input and output (I/O), the Unix file model worked quite well as most I/O was "linear". However, modern systems include networking and other new devices. Describing a graphical user interface driven by mouse control in an "event driven" fashion didn't work well under the old model. Work on systems supporting these new devices in the 1980s led to facilities for non-blocking I/O, forms of inter-process communications other than just pipes, as well as moving functionality such as network protocols out of the kernel.

41.2 History

A partial list of simultaneously running processes on a Unix system.

In the 1960s, the Massachusetts Institute of Technology, AT&T Bell Labs, and General Electric worked on an experimental operating system called Multics (**Mult**iplexed **I**nformation and **C**omputing **S**ervice), which was designed to run on the GE-645 mainframe computer. The aim was the creation of a commercial product, although this was never a great success. Multics was an interactive operating system with many novel capabilities, including enhanced security. The project did develop production releases, but initially these releases performed poorly.

AT&T Bell Labs pulled out and deployed its resources elsewhere. One of the developers on the Bell Labs team, Ken Thompson, continued to develop for the GE-645 mainframe, and wrote a game for that computer called Space Travel. However, he found that the game was too slow on the GE machine and was expensive, costing $75 per execution in scarce computing time.

Thompson thus re-wrote the game in assembly language for Digital Equipment Corporation's PDP-7 with help from Dennis Ritchie. This experience, combined with his work on the Multics project, led Thompson to start a new operating system for the PDP-7. Thompson and Ritchie led a team of developers, including Rudd Canaday, at Bell Labs developing a file system as well as the new multi-tasking operating system itself. They included a command line interpreter and some small utility programs.

Editing a shell script using the *ed* editor. The dollar-sign at the top of the screen is the prompt printed by the shell. 'ed' is typed to start the editor, which takes over from that point on the screen downwards.

41.2.1 1970s

In 1970 the project was named **Unics**, and could - eventually - support two simultaneous users. Brian Kernighan invented this name as a contrast to **Multics**; the spelling was later changed to **Unix**.

Up until this point there had been no financial support from Bell Labs. When the Computer Science Research Group wanted to use Unix on a much larger machine than the PDP-7, Thompson and Ritchie managed to trade the promise of adding text processing capabilities to Unix for a PDP-11/20 machine. This led to some financial support from Bell. For the first time in 1970, the Unix operating system was officially named and ran on the PDP-11/20. It added a text formatting program called roff and a text editor. All three were written in PDP-11/20 assembly language. Bell

Labs used this initial "text processing system", made up of Unix, roff, and the editor, for text processing of patent applications. Roff soon evolved into troff, the first electronic publishing program with a full typesetting capability. The *UNIX Programmer's Manual* was published on November 3, 1971.

In 1973, Unix was rewritten in the C programming language, contrary to the general notion at the time "that something as complex as an operating system, which must deal with time-critical events, had to be written exclusively in assembly language" [4]. The migration from assembly language to the higher-level language C resulted in much more portable software, requiring only a relatively small amount of machine-dependent code to be replaced when porting Unix to other computing platforms.

AT&T made Unix available to universities and commercial firms, as well as the United States government under licenses. The licenses included all source code including the machine-dependent parts of the kernel, which were written in PDP-11 assembly code. Copies of the annotated Unix kernel sources circulated widely in the late 1970s in the form of a much-copied book by John Lions of the University of New South Wales, the *Lions' Commentary on UNIX 6th Edition, with Source Code*, which led to considerable use of Unix as an educational example.

Versions of the Unix system were determined by editions of its user manuals, so that (for example) "Fifth Edition UNIX" and "UNIX Version 5" have both been used to designate the same thing. Development expanded, with Versions 4, 5, and 6 being released by 1975. These versions added the concept of pipes, leading to the development of a more modular code-base, increasing development speed still further. Version 5 and especially Version 6 led to a plethora of different Unix versions both inside and outside Bell Labs, including PWB/UNIX, IS/1 (the first commercial Unix), and the University of Wollongong's port to the Interdata 7/32 (the first non-PDP Unix).

In 1978, UNIX/32V, for the VAX system, was released. By this time, over 600 machines were running Unix in some form. Version 7 Unix, the last version of Research Unix to be released widely, was released in 1979. Versions 8, 9 and 10 were developed through the 1980s but were only released to a few universities, though they did generate papers describing the new work. This research led to the development of Plan 9 from Bell Labs, a new portable distributed system.

41.2.2 1980s

A late-80s style Unix desktop running the *X Window System* graphical user interface. Shown are a number of client applications common to the MIT X Consortium's distribution, including Tom's Window Manager, an X Terminal, Xbiff, xload, and a graphical manual page browser.

AT&T now licensed UNIX System III, based largely on Version 7, for commercial use, the first version launching in 1982. This also included support for the VAX. AT&T continued to issue licenses for older Unix versions. To end the confusion between all its differing internal versions, AT&T combined them into UNIX System V Release 1. This introduced a few features such as the vi editor and curses from the Berkeley Software Distribution of Unix developed at the University of California, Berkeley. This also included support for the Western Electric 3B series of machines.

Since the newer commercial UNIX licensing terms were not as favorable for academic use as the older versions of Unix, the Berkeley researchers continued to develop BSD Unix as an alternative to UNIX System III and V, originally on the PDP-11 architecture (the 2.xBSD releases, ending with 2.11BSD) and later for the VAX-11 (the 4.x BSD releases). Many contributions to Unix first appeared on BSD systems, notably the C shell with job control (modelled on ITS). Perhaps the most important aspect of the BSD development effort was the addition of TCP/IP network code to the mainstream Unix kernel. The BSD effort produced several significant releases that contained network code: 4.1cBSD, 4.2BSD, 4.3BSD, 4.3BSD-Tahoe ("Tahoe" being the nickname of the Computer Consoles Inc. Power 6/32 architecture that was the first non-DEC release of the BSD kernel), Net/1, 4.3BSD-Reno (to match the "Tahoe" naming, and that the release was something of a gamble), Net/2, 4.4BSD, and 4.4BSD-lite. The network code found in these releases is the ancestor of much TCP/IP network code in use today, including code that was later released in AT&T System V UNIX and early versions of Microsoft Windows. The accompanying

Berkeley Sockets API is a de facto standard for networking APIs and has been copied on many platforms.

Other companies began to offer commercial versions of the UNIX System for their own mini-computers and workstations. Most of these new Unix flavors were developed from the System V base under a license from AT&T; however, others were based on BSD instead. One of the leading developers of BSD, Bill Joy, went on to co-found Sun Microsystems in 1982 and create SunOS (now Solaris) for their workstation computers. In 1980, Microsoft announced its first Unix for 16-bit microcomputers called Xenix, which the Santa Cruz Operation (SCO) ported to the Intel 8086 processor in 1983, and eventually branched Xenix into SCO UNIX in 1989.

For a few years during this period (before PC compatible computers with MS-DOS became dominant), industry observers expected that UNIX, with its portability and rich capabilities, was likely to become the industry standard operating system for microcomputers.[5] In 1984 several companies established the X/Open consortium with the goal of creating an open system specification based on UNIX. Despite early progress, the standardization effort collapsed into the "Unix wars," with various companies forming rival standardization groups. The most successful Unix-related standard turned out to be the IEEE's POSIX specification, designed as a compromise API readily implemented on both BSD and System V platforms, published in 1988 and soon mandated by the United States government for many of its own systems.

AT&T added various features into UNIX System V, such as file locking, system administration, streams, new forms of IPC, the Remote File System and TLI. AT&T cooperated with Sun Microsystems and between 1987 and 1989 merged features from Xenix, BSD, SunOS, and System V into System V Release 4 (SVR4), independently of X/Open. This new release consolidated all the previous features into one package, and heralded the end of competing versions. It also increased licensing fees.

During this time a number of vendors including Digital Equipment, Sun, Addamax and others began building trusted versions of UNIX for high security applications, mostly designed for military and law enforcement applications.

The Common Desktop Environment or CDE, a graphical desktop for Unix co-developed in the 1990s by HP, IBM, and Sun as part of the COSE initiative.

41.2.3 1990s

In 1990, the Open Software Foundation released OSF/1, their standard Unix implementation, based on Mach and BSD. The Foundation was started in 1988 and was funded by several Unix-related companies that wished to counteract the collaboration of AT&T and Sun on SVR4. Subsequently, AT&T and another group of licensees formed the group "UNIX International" in order to counteract OSF. This escalation of conflict between competing vendors gave rise again to the phrase "Unix wars".

In 1991, a group of BSD developers (Donn Seeley, Mike Karels, Bill Jolitz, and Trent Hein) left the University of California to found Berkeley Software Design, Inc (BSDI). BSDI produced a fully functional commercial version of BSD Unix for the inexpensive and ubiquitous Intel platform, which started a wave of interest in the use of inexpensive hardware for production computing. Shortly after it was founded, Bill Jolitz left BSDI to pursue distribution of 386BSD, the free software ancestor of FreeBSD, OpenBSD, and NetBSD.

By 1993 most commercial vendors had changed their variants of Unix to be based on System V with many BSD features added on top. The creation of the COSE initiative that year by the major players in Unix marked the end of the most notorious phase of the Unix wars, and was followed by the merger of UI and OSF in 1994. The new combined entity, which retained the OSF name, stopped work on OSF/1 that year. By that time the only vendor using it was Digital, which continued its own development, rebranding their product Digital UNIX in early 1995.

Shortly after UNIX System V Release 4 was produced, AT&T sold all its rights to UNIX® to Novell. (Dennis Ritchie likened this to the Biblical story of Esau selling his birthright for the proverbial "mess of pottage".[6]) Novell developed its own version, UnixWare, merging its NetWare

with UNIX System V Release 4. Novell tried to use this to battle against Windows NT, but their core markets suffered considerably.

In 1993, Novell decided to transfer the UNIX® trademark and certification rights to the X/Open Consortium.[7] In 1996, X/Open merged with OSF, creating the Open Group. Various standards by the Open Group now define what is and what is not a "UNIX" operating system, notably the post-1998 Single UNIX Specification.

In 1995, the business of administering and supporting the existing UNIX licenses, plus rights to further develop the System V code base, were sold by Novell to the Santa Cruz Operation.[1] Whether Novell also sold the copyrights is currently the subject of litigation (see below).

In 1997, Apple Computer sought out a new foundation for its Macintosh operating system and chose NEXTSTEP, an operating system developed by NeXT. The core operating system was renamed Darwin after Apple acquired it. It was based on the BSD family and the Mach kernel. The deployment of Darwin BSD Unix in Mac OS X makes it, according to a statement made by an Apple employee at a USENIX conference, the most widely used Unix-based system in the desktop computer market.

41.2.4 2000 to present

Fig. 8. A modern Unix desktop environment (Solaris 10)

In 2000, SCO sold its entire UNIX business and assets to Caldera Systems, which later on changed its name to The SCO Group. This new player then started legal action against various users and vendors of Linux. SCO have alleged that Linux contains copyrighted Unix code now owned by The SCO Group. Other allegations include trade-secret violations by IBM, or contract violations by former Santa Cruz customers who have since converted to Linux. However, Novell disputed the SCO group's claim to hold copyright on the UNIX source base. According to Novell, SCO (and hence the SCO Group) are effectively franchise operators for Novell, which also retained the core copyrights, veto rights over future licensing activities of SCO, and 95% of the licensing revenue. The SCO Group disagreed with this, and the dispute had resulted in the SCO v. Novell lawsuit.

In 2005, Sun Microsystems released the bulk of its Solaris system code (based on UNIX System V Release 4) into an open source project called OpenSolaris. New Sun OS technologies such as the ZFS file system are now first released as open source code via the OpenSolaris project; as of 2006 it has spawned several non-Sun distributions such as SchilliX, Belenix, Nexenta and MarTux.

The Dot-com crash has led to significant consolidation of Unix users as well. Of the many commercial flavors of Unix that were born in the 1980s, only Solaris, HP-UX, and AIX are still doing relatively well in the market, though SGI's IRIX persisted for quite some time. Of these, Solaris has the most market share, and may be gaining popularity due to its feature set and also since it now has an Open Source version.

41.3 Standards

Beginning in the late 1980s, an open operating system standardization effort now known as POSIX provided a common baseline for all operating systems; IEEE based POSIX around the common structure of the major competing variants of the Unix system, publishing the first POSIX standard in 1988. In the early 1990s a separate but very similar effort was started by an industry consortium, the Common Open Software Environment (COSE) initiative, which eventually became the Single UNIX Specification administered by The Open Group). Starting in 1998 the Open Group and IEEE started the Austin Group, to provide a common definition of POSIX and the Single UNIX Specification.

In an effort towards compatibility, in 1999 several Unix system vendors agreed on SVR4's Executable and Linkable Format (ELF) as the standard for binary and object code files. The common format allows substantial binary compatibility among Unix systems operating on the same CPU architecture.

The Filesystem Hierarchy Standard was created to provide a reference directory layout for Unix-like operating systems, particularly Linux. This type of standard however is controversial, and even within the Linux community its adoption is far from universal.

41.4 Components

The Unix system is composed of several components that are normally packaged together. By including — in addition to the kernel of an operating system — the development environment, libraries, documents, and the portable, modifiable source-code for all of these components, Unix was a self-contained software system. This was one of the key reasons it emerged into an important teaching and learning tool and had such a broad influence.

Inclusion of these components did not make the system large — the original V7 UNIX distribution, consisting of copies of all of the compiled binaries plus all of the source code and documentation occupied less than 10Mb, and arrived on a single 9-track magtape. The printed documentation, typeset from the on-line sources, was contained in two volumes.

The names and filesystem locations of the Unix components has changed substantially across the history of the system. Nonetheless, the V7 implementation is considered by many to have the canonical early structure:

- **Kernel** — source code in /usr/sys, composed of several sub-components:
  - *conf* — configuration and machine-dependent parts, including boot code
  - *dev* — device drivers for control of hardware (and some pseudo-hardware)
  - *sys* — operating system "kernel", handling memory management, process scheduling, system calls, etc.
  - *h* — header files, defining key structures within the system and important system-specific invariables
- **Development Environment** — Early versions of Unix contained a development environment sufficient to recreate the entire system from source code:
  - *cc* — C language compiler (first appeared in V3 Unix)
  - *as* — machine-language assembler for the machine
  - *ld* — linker, for combining object files
  - *lib* — object-code libraries (installed in /lib or /usr/lib) *libc*, the system library with C run-time support, was the primary library, but there have always been additional libraries for such things as mathematical functions (*libm*) or database access. V7 Unix introduced the first version of the modern "Standard I/O" library *stdio* as part of the system library. Later implementations increased the number of libraries significantly.
  - *make* - build manager (introduced in PWB/UNIX), for effectively automating the build process
  - *include* — header files for software development, defining standard interfaces and system invariants
  - *Other languages* — V7 Unix contained a Fortran-77 compiler, a programmable arbitrary-precision calculator (*bc*, *dc*), and the awk "scripting" language, and later

versions and implementations contain many other language compilers and toolsets. Early BSD releases included Pascal tools, and many modern Unix systems also include the GNU Compiler Collection as well as or instead of a proprietary compiler system.
    - *Other tools* — including an object-code archive manager (*ar*), symbol-table lister (*nm*), compiler-development tools (e.g. *lex* & *yacc*), and debugging tools.
- **Commands** — Unix makes little distinction between commands (user-level programs) for system operation and maintenance (e.g. *cron*), commands of general utility (e.g. *grep*), and more general-purpose applications such as the text formatting and typesetting package. Nonetheless, some major categories are:
    - *sh* — The "shell" programmable command-line interpreter, the primary user interface on Unix before window systems appeared, and even afterward (within a "command window").
    - *Utilities* — the core tool kit of the Unix command set, including *cp*, *ls*, *grep*, *find* and many others. Subcategories include:
        - *System utilities* — administrative tools such as *mkfs*, *fsck*, and many others
        - *User utilities* — environment management tools such as *passwd*, *kill*, and others.
    - *Document formatting* — Unix systems were used from the outset for document preparation and typesetting systems, and included many related programs such as *nroff*, *troff*, *tbl*, *eqn*, *refer*, and *pic*. Some modern Unix systems also include packages such as TeX and GhostScript.
    - *Graphics* — The *plot* subsystem provided facilities for producing simple vector plots in a device-independent format, with device-specific interpreters to display such files. Modern Unix systems also generally include X11 as a standard windowing system and GUI, and many support OpenGL.
    - *Communications* — Early Unix systems contained no inter-system communication, but did include the inter-user communication programs *mail* and *write*. V7 introduced the early inter-system communication system *UUCP*, and systems beginning with BSD release 4.1c included TCP/IP utilities.

The 'man' command can display a 'man page' for every command on the system, including itself.

- **Documentation** — Unix was the first operating system to include all of its documentation online in machine-readable form. The documentation included:
    - *man* — manual pages for each command, library component, system call, header file, etc.
    - *doc* — longer documents detailing major subsystems, such as the C language and *troff*

41.5 Impact

The Unix system had significant impact on other operating systems.

It was written in high level language as opposed to assembly language (which had been thought necessary for systems implementation on early computers). Although this followed the lead of Multics and Burroughs, it was Unix that popularized the idea.

Unix had a drastically simplified file model compared to many contemporary operating systems, treating all kinds of files as simple byte arrays. The file system hierarchy contained machine services and devices (such as printers, terminals, or disk drives), providing a uniform interface, but at the expense of occasionally requiring additional mechanisms such as ioctl and mode flags to access features of the hardware that did not fit the simple "stream of bytes" model. The Plan 9 operating system pushed this model even further and eliminated the need for additional mechanisms.

Unix also popularized the hierarchical file system with arbitrarily nested subdirectories, originally introduced by Multics. Other common operating systems of the era had ways to divide a storage device into multiple directories or sections, but they had a fixed number of levels, often only

one level. Several major proprietary operating systems eventually added recursive subdirectory capabilities also patterned after Multics. DEC's RSX-11M's "group, user" hierarchy evolved into VMS directories, CP/M's volumes evolved into MS-DOS 2.0+ subdirectories, and HP's MPE group.account hierarchy and IBM's SSP and OS/400 library systems were folded into broader POSIX file systems.

Making the command interpreter an ordinary user-level program, with additional commands provided as separate programs, was another Multics innovation popularized by Unix. The Unix shell used the same language for interactive commands as for scripting (shell scripts — there was no separate job control language like IBM's JCL). Since the shell and OS commands were "just another program", the user could choose (or even write) his own shell. New commands could be added without changing the shell itself. Unix's innovative command-line syntax for creating chains of producer-consumer processes (pipelines) made a powerful programming paradigm (coroutines) widely available. Many later command-line interpreters have been inspired by the Unix shell.

A fundamental simplifying assumption of Unix was its focus on ASCII text for nearly all file formats. There were no "binary" editors in the original version of Unix — the entire system was configured using textual shell command scripts. The common denominator in the I/O system is the byte — unlike "record-based" file systems in other computers. The focus on text for representing nearly everything made Unix pipes especially useful, and encouraged the development of simple, general tools that could be easily combined to perform more complicated *ad hoc* tasks. The focus on text and bytes made the system far more scalable and portable than other systems. Over time, text-based applications have also proven popular in application areas, such as printing languages (PostScript), and at the application layer of the Internet Protocols, e.g. Telnet, FTP, SSH, SMTP, HTTP and SIP.

Unix popularised a syntax for regular expressions that found widespread use. The Unix programming interface became the basis for a widely implemented operating system interface standard (POSIX, see above).

The C programming language soon spread beyond Unix, and is now ubiquitous in systems and applications programming.

Early Unix developers were important in bringing the theory of modularity and reusability into software engineering practice, spawning a "Software Tools" movement.

Unix provided the TCP/IP networking protocol on relatively inexpensive computers, which contributed to the Internet explosion of world-wide real-time connectivity, and which formed the basis for implementations on many other platforms. (This also exposed numerous security holes in the networking implementations.)

The Unix policy of extensive on-line documentation and (for many years) ready access to all system source code raised programmer expectations, contributed to the 1983 launch of the free software movement.

Over time, the leading developers of Unix (and programs that ran on it) evolved a set of cultural norms for developing software, norms which became as important and influential as the technology of Unix itself; this has been termed the Unix philosophy.

Unix stores system time values as the number of seconds from midnight January 1, 1970 (the "Unix Epoch") in variables of type time_t, historically defined as "signed 32-bit integer". On January 19, 2038, the current time will roll over from a zero followed by 31 ones (01111111111111111111111111111111) to a one followed by 31 zeros (10000000000000000000000000000000), which will reset time to the year 1901 or 1970, depending on implementation, because that toggles the sign bit. As many applications use OS library routines for date calculations, the impact of this could be felt much earlier than 2038; for instance, 30-year mortgages may be calculated incorrectly beginning in the year 2008.

Since times before 1970 are rarely represented in Unix time, one possible solution that is compatible with existing binary formats would be to redefine time_t as "unsigned 32-bit integer". However, such a kludge merely postpones the problem to February 7, 2106, and could introduce bugs in software that compares differences between two sets of time.

Some Unix versions have already addressed this. For example, in Solaris on 64-bit systems, time_t is 64 bits long, meaning that the OS itself and 64-bit applications will correctly handle dates for some 292 billion years (several times greater than the age of the universe). Existing 32-bit applications using a 32-bit time_t continue to work on 64-bit Solaris systems but are still prone to the 2038 problem.

41.6 Free Unix-like operating systems

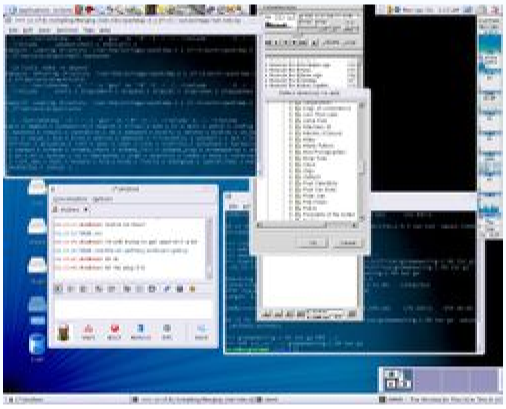


Fig. 9. Linux is a modern Unix-like system

In 1983, Richard Stallman announced the GNU project, an ambitious effort to create a free software Unix-like system; "free" in that everyone who received a copy would be free to use, study, modify, and redistribute it. GNU's goal was achieved in 1992. Its own kernel development project, GNU Hurd, had not produced a working kernel, but a compatible kernel called Linux was released as free software in 1992 under the GNU General Public License. The combination of the two is frequently referred to simply as "Linux", although the Free Software Foundation and some Linux distributions, such as Debian GNU/Linux, use the combined term GNU/Linux. Work on GNU Hurd continues, although very slowly.

In addition to their use in the Linux operating system, many GNU packages — such as the GNU Compiler Collection (and the rest of the GNU toolchain), the GNU C library and the GNU core utilities — have gone on to play central roles in other free Unix systems as well.

Linux distributions, comprising Linux and large collections of compatible software have become popular both with hobbyists and in business. Popular distributions include Red Hat Enterprise Linux, SUSE Linux, Mandriva Linux, Fedora, Ubuntu, Debian GNU/Linux, Slackware Linux and Gentoo.

A free derivative of BSD Unix, 386BSD, was also released in 1992 and led to the NetBSD and FreeBSD projects. With the 1994 settlement of a lawsuit that UNIX Systems Laboratories brought against the University of California and Berkeley Software Design Inc. (USL v. BSDi), it was clarified that Berkeley had the right to distribute BSD Unix — for free, if it so desired. Since then, BSD Unix has been developed in several different directions, including the OpenBSD and DragonFly BSD variants.

Linux and the BSD kin are now rapidly occupying the market traditionally occupied by proprietary Unix operating systems, as well as expanding into new markets such as the consumer desktop and mobile and embedded devices. A measure of this success may be seen when Apple Computer incorporated BSD into its Macintosh operating system by way of NEXTSTEP. Due to the modularity of the Unix design, sharing bits and pieces is relatively common; consequently, most or all Unix and Unix-like systems include at least some BSD code, and modern BSDs also typically include some GNU utilities in their distribution, so Apple's combination of parts from NeXT and FreeBSD with Mach and some GNU utilities has precedent.

In 2005, Sun Microsystems released the bulk of the source code to the Solaris operating system, a System V variant, under the name OpenSolaris, making it the first actively developed commercial Unix system to be open sourced (several years earlier, Caldera had released many of the older Unix systems under an educational and later BSD license). As a result, a great deal of formerly proprietary AT&T/USL code is now freely available.

41.7 Branding

In October 1993, Novell, the company that owned the rights to the Unix System V source at the time, transferred the trademarks of Unix to the X/Open Company (now The Open Group),[9] and in 1995 sold the related business operations to Santa Cruz Operation.[10] Whether Novell also sold the copyrights to the actual software is currently the subject of litigation in a federal lawsuit, SCO v. Novell. Unix vendor SCO Group Inc. accused Novell of slander of title.

The present owner of the trademark *UNIX*® is The Open Group, an industry standards consortium. Only systems fully compliant with and certified to the Single UNIX Specification qualify as "UNIX®" (others are called "Unix system-like" or "Unix-like"). The term UNIX is not an acronym, but follows the early convention of naming computer systems in capital letters, such as ENIAC and MISTIC.

By decree of The Open Group, the term "UNIX®" refers more to a class of operating systems than to a specific implementation of an operating system; those operating systems which meet The Open Group's Single UNIX Specification should be able to bear the UNIX® 98 or UNIX® 03 trademarks today, after the operating system's vendor pays a fee to The Open Group. Systems licensed to use the UNIX® trademark include AIX, HP-UX, IRIX, Solaris, Tru64, A/UX, Mac OS X 10.5 on Intel platforms[11], and a part of z/OS.

Sometimes a representation like "Un*x", "*NIX", or "*N?X" is used to indicate all operating systems similar to Unix. This comes from the use of the "*" and "?" characters as "wildcard" characters in many utilities. This notation is also used to describe other Unix-like systems, e.g. Linux, FreeBSD, etc., that have not met the requirements for UNIX® branding from the Open Group.

The Open Group requests that "UNIX®" is always used as an adjective followed by a generic term such as "system" to help avoid the creation of a genericized trademark.

The term "Unix" is also used, and in fact was the original capitalisation, but the name UNIX stuck because, in the words of Dennis Ritchie "when presenting the original Unix paper to the third Operating Systems Symposium of the American Association for Computing Machinery, we had just acquired a new typesetter and were intoxicated by being able to produce small caps" (quoted from the Jargon File, version 4.3.3, 20 September 2002). Additionally, it should be noted that many of the operating system's predecessors and contemporaries used all-uppercase lettering, because many computer terminals of the time could not produce lower-case letters, so many people wrote the name in upper case due to force of habit.

Several plural forms of Unix are used to refer to multiple brands of Unix and Unix-like systems. Most common is the conventional "**Unixes**", but the hacker culture which created Unix has a penchant for playful use of language, and "**Unices**" (treating Unix as Latin noun of the third declension) is also popular. The Anglo-Saxon plural form "Unixen" is not common, although occasionally seen.

Trademark names can be registered by different entities in different countries and trademark laws in some countries allow the same trademark name to be controlled by two different entities if each entity uses the trademark in easily distinquishable categories. The result is that Unix has been used as a brand name for various products including book shelves, ink pens, bottled glue, diapers, hair driers and food containers. [2].

41.8 Common Unix commands

Widely used Unix commands include:

- Directory and file creation and navigation: *ls cd pwd mkdir rm rmdir cp find touch*

- File viewing and editing: *more less ed vi emacs head tail*
- Text processing: *echo cat grep sort uniq sed awk cut tr split printf*
- File comparison: *comm cmp diff patch*
- Miscellaneous shell tools: *yes test xargs*
- System administration: *chmod chown ps su w who*
- Communication: *mail telnet ftp finger ssh*
- Authentication: *su login passwd*

42 OS/2 OPERATING SYSTEM

In the early 1990s, two of the biggest names in the PC world, IBM and Microsoft, joined forces to create OS/2, with the goal of making it the "next big thing" in graphical operating systems. Well, it didn't quite work out that way. :^) The story behind OS/2 includes some of the most fascinating bits of PC industry history, but it's a long story and not one that really makes sense to get into here. The short version goes something like this:

Microsoft and IBM create OS/2 with high hopes that it will revolutionize the PC desktop.

OS/2 has some significant technical strengths but also some problems.

Microsoft and IBM fight over how to fix the problems, and also over what direction to take for the future of the operating system.

Microsoft decides, based on some combination of frustration over problems and desire for absolute control, to drop OS/2 and focus on Windows instead.

IBM and Microsoft feud.

IBM supports OS/2 (somewhat half-heartedly) on its own, while Microsoft dominates the industry with various versions of Windows.

Now, OS/2 aficionados will probably take issue with at least some of that summarization, but that is what happened in a nutshell, or at least I think so. :^) At any rate, OS/2 continues to be supported today, but really has been relegated to a niche role. I don't know how long IBM will continue to support it.

OS/2's file system support is similar, in a way to that of Windows NT's. OS/2 supports FAT12 and FAT16 for compatibility, but is really designed to use its own special file system, called HPFS. HPFS is similar to NTFS (NT's native file system) though it is certainly not the same. OS/2 does not have support for FAT32 built in, but that there are third-party tools available that will let OS/2 access FAT32 partitions. This may be required if you are running a machine with both OS/2 and Windows partitions. I believe that OS/2 does not include support for NTFS partitions.

43 References

1. Ричард Петерсен .“LINUX: руководство по операционной системе
2. D.Azzarito, David W. Green (Contributor). “The Os/2 Warp Survival Guide: Installing, Configuring, and Using Os/2 2.X”
3. G.Harvey. “Dos for Dummies Quick Reference”
4. J.Duntemann. “Assembly Language Step-by-step: Programming with DOS and Linux“
5. K.August. “Social Indicators and Social Theory: Elements of an Operational System”
6. A.Slater. “Handbook of Physical Distribution Software: Distribution Planning and Operational Systems”
7. В.Э.Фигурнов. "IBM PC для пользователя 6 – е издание". Москва, 1995
8. Панкратов Е. “'Операционная система MS-DOS 6.22: Справочное пособие”
9. A.Vidžiūnas, D.Vitkutė. “Personalinių kompiuterių operacinės sistemos”. Vilnius, “Mokslo Aidas”, 1995
10. М. Гук. “Аппаратные средства IBM PC”. Санкт – Петербург, 2001
11. J.L.Hennessy, A.A.Patterson. “Compiuter Architecture: A Quantitative Approach.” San Mateo, 1996
12. А.Робачевский “Операционная система UNIX”
13. Б.Керниган, Р.Пайк “UNIX. Програмное окружение”
14. Mark T. Chapman “Os/2 Power User's Reference: From Os/2 2.0 Through Warp”, 1996
15. A. Venčkauskas, A.Venčkauskienė “Operacinių sistemų pradmenys”, Kaunas, “Technologija”, 2002
16. http://web.mit.edu/invent/iow/russell.html
17. http://seattletimes.nwsource.com/html/businesstechnology/2002103322_cdman29.html
18. http://inventors.about.com/library/inventors/blopticaldisk.htm
19. http://en.wikipedia.org/wiki/Unix